(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
16 January 2003 (16.01.2003)

PCT

(10) International Publication Number
WO 03/005184 A2

(51) International Patent Classification[7]: G06F 9/00

(21) International Application Number: PCT/IB01/01186

(22) International Filing Date: 4 July 2001 (04.07.2001)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for all designated States except US)*: **AKODI S.A.** [CH/CH]; Chemin du Pavillon 2, CH-1218 Grand-Saconnex (CH).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **LEROY, Eric** [CH/CH]; 9, rue Cavour, CH-1203 Geneva (CH). **TETTONI, Laurent** [CH/CH]; Chemin de la Gradelle 71, CH-1224 Chene-Bougeries (CH).

(74) Agent: **MICHELI & CIE**; 122, rue de Genève, Case postale 61, CH-1226 Thonex (CH).

(81) Designated States *(national)*: AE, AG, AL, AM, AT (utility model), AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ (utility model), CZ, DE (utility model), DE, DK (utility model), DK, DM, DZ, EC, EE (utility model), EE, ES, FI (utility model), FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK (utility model), SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

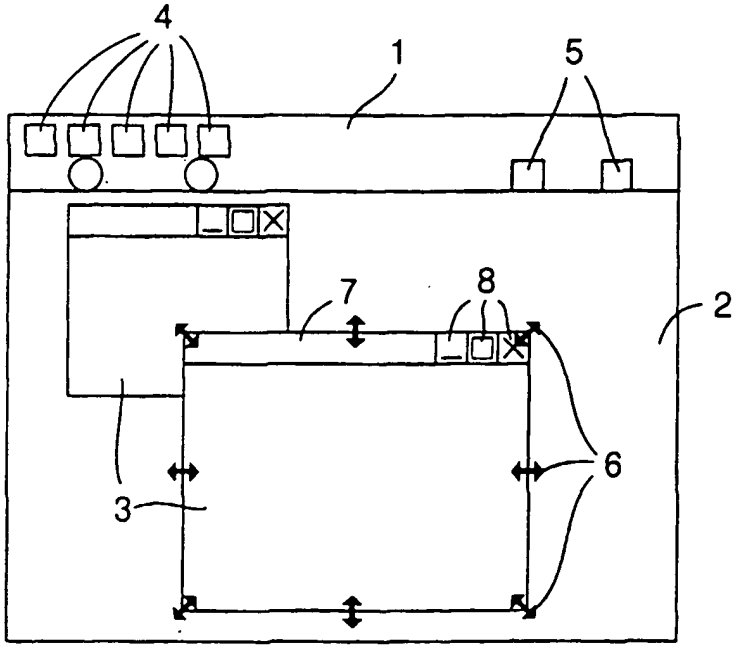
(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: WEB WINDOWED GRAPHICAL USER INTERFACE



(57) Abstract: A method for providing a windowed graphical user interface within a single Web Page. Thanks to the use of Dynamic HTML divisions and Java applets, a single web page downloaded from a web server, displays a desktop page where individual windows can be created and managed. A window manager is set up when the desktop page is downloaded and allows synchronization of the WebWindows applets with the divisions of the DHTML page. The invention has also for object a graphical user interface.

WO 03/005184 A2

Web windowed graphical user interface

The present invention relates to information presentation within a standard Web browser and more particularly to a method for displaying and managing web applications executed in independent windows within a standard HTML web page. The invention has also for object a windowed graphical user interface within an HTML page.

A portion of the disclosure of this patent document contains material which is subject to copyright protection and to which a claim of copyright protection is made. The copyright owner has no objection to the facsimile reproduction by anyone of the patent disclosure, as it appears in the Patent and Trademark Office patent files or records, but otherwise reserves all copyrights and similar rights whatsoever

The Internet World Wide Web has become a major platform for Business-to-Consumer and Business-to-Business exchange. Numerous servers for data or applications provide information or services to millions of potential customers through a dense network of interconnected computers.

Internet extends the reach of companies, and simplifies data and application deployment, since standard exists on the client side, namely the Internet web browser. Web browser programs are standardized in the sense that such software complies with a normalized specification. Nowadays, the market is mainly shared by Microsoft's Internet Explorer™ and Netscape's Communicator™, both products are distributed free of charge.

For these reasons, software vendors now work at enabling their existing applications for the web. Obviously, most new software are web-enabled, since the Internet inherently allows graphical, interactive, authenticated, and secure remote sessions.

Meanwhile, users demand for an increasing functionality. The web is no more a place to search for informational pages; it has become the place for online

transactions, for operating complex applications that involve document retrieval, database access, e-commerce, content or news publication, etc. An increased complexity and increased traffic require improved human/machine interfaces, still by presenting the information on the end-user's browser. The browser is not versatile enough for such complex functionality; it has become a constraint.

Interestingly, web sites evolved from simple text content, to sophisticated menu-driven navigation sites, still page-based, that we can see now. Yet, there are no window-based applications available on the Web, nor software development kits to allow their easy development. It is therefore an object of the present invention to provide such a windowed environment that can be used within a standard browser application.

Computer science engineers are faced with the problem of designing and implementing systems to collect, transform and deliver larger amounts of data, in more complex form. Over the last years, the prominent model for such systems is the "multi-tier" architecture, where data is centralized into databases; applications and services are distributed across redundant application servers; and the interactive part is deployed to individual users in its lightest form.

Most multi-tier architectures are nowadays implemented over the Web (either Internet or intranet), where a standard client program operates the client tier: the Browser. Thus, maintenance and deployment costs are almost cancelled, since there is no specific software to install and maintain on potentially millions of users workstations.

Unfortunately, this economic view imposes dramatic constraints on the type of interactivity that can be proposed to the user. Regular Web sites are generally presented as a series of pages, plus a navigation system that persists on every page, to allow switching to other pages. This is clearly not flexible enough when both dense information and complex services are to be provided to the user. For instance, it is impossible to visualize two pages simultaneously.

One object of the invention is to provide a light, yet flexible and versatile platform that would allow the browser to display various kinds of information and services provided by a server. The platform would present a unity of look and feel, of behavior, and of course a single sign-on (same user, authentication, security and session for all information and services).

The maintenance of complex web sites is also problematic. When a new content source or when a new service is added, it impacts on the global navigation system, and it might require cross-reference links to related information pages or services. Similarly, when some content or interactive form is modified, it might impact on the whole page formatting, because of the page-view mode.

In rich web sites like portals, or companies Intranets, the navigation is known to raise difficulties. For this reason, "site maps" are sometimes available.

From the user's standpoint, too much information is displayed on every page, and the personalization, when available, is rather difficult and unintuitive. The very fact that the navigation is page-based questions what should appear on the page, when, and where.

From the site developer's standpoint, the aggregation of images, content, and searching into a single page is a challenging task. It requires specialized software, usually complex and expensive. For instance, Content Delivery Software is usually a component of Electronic Content Management solutions but it requires graphical and programming skills, for aggregating content from different origins in to a nice-looking browser page.

Another drawback of classical web sites is their difficulty to integrate real applications, as soon as their graphical user interface goes beyond classical forms, or as soon as they need immediate access to the server for computing purposes.

Using a windowing system would allow a neat integration of such applications within the existing system.

In the framework that is proposed here, the Web browser becomes like a regular terminal with a windowing user interface: the user may easily access the information or application required to perform his/her task, and the Web site developer obtains a clear separation of all those windows, since each is a defined application.

Regular operating systems use the metaphor of the desktop to present information and services that can be viewed and accessed in independent windows. This metaphor is widely accepted and successful.

A goal of the present invention is to implement this architecture on client-server systems based on the Web model. In other word, this means that a windowing system is implemented at the browser level thanks to the method object of the invention.

The aim of the present invention is to provide a windowed graphical user interface that can be displayed in a Web page thanks to existing web browser applications. This user interface will allow a user to arrange independent windows in a personal arrangement, to facilitate the display of information. Windows can be positioned relative to each other, also allowing viewing several windows simultaneously.

These goals are achieved thanks to a method having the characteristics recited in claim 1 and a graphical user interface according to claim 7.

Thanks to this method, the following advantages are achieved.

The display of web pages is no longer page-oriented. The user can personalize its working environment by selecting which information is presented and where it is displayed. Windows can be temporarily minimized, maximized or overlaid. Furthermore, any request pertaining to a particular window does not affect the whole display, or the other windows.

As the desktop metaphor is now commonly used and understood, there is no required training period for using the graphical user interface object of the invention.

Lastly, thanks to the use of standard software libraries, all applications can be designed with similar aspect and behavior, in a more flexible manner than by using regular HTML pages.

The invention will now be described in details in the following specification and with reference to the accompanying drawings in which:

Figure 1 is a schematic representation of the desktop presented to the user with two WebWindows.

Figure 2 is a schematic representation of the components downloaded and executed within the browser as well as their data storage.

Figure 3 is a schematic view representing the overlaying of Java and DHTML objects within the browser.

Figure 4 is a schematic view of the class tree structure.

On the World Wide Web, the level of freedom granted to site developers is limited by the normalized specification of how the browser on the client side interacts with the server thanks to the hypertext transfer protocol (thereafter abbreviated HTTP), and how the content is presented on the user's screen thanks to the normalized page-description language, the hypertext markup language (thereafter abbreviated HTML).

The success of the World Wide Web rapidly called for improved graphical and interactive functionality, whereas the underlying HTTP protocol remained stable. Indeed, HTML evolved from version 1 to currently four, with major functionality being added. Those required for the present invention will be briefly described in the in the following sections.

HTML was meant to be a page-description language, for text documents embedding tagged entities like presentation styles, paragraph formatting rules, section numbering, remote links and anchors; but it quickly evolved to hypermedia allowing the display of complex documents including images, sounds, and movies.

The next progress was scripting, which allows the execution of complex processing at the client's browser side, in order to refine the presentation, or to animate it, among others. Thanks to scripts, pages can be made dynamic and reactive to the user's behavior. This feature is fundamental to any graphical user interface, that could not be designed on static pages, whatever their complexity. The most popular browser scripting language, JavaScript, is a component required for implementing the method according to the invention.

Any newspaper or content-rich source is graphically structured with titles, columns, sections, paragraphs, etc. Whereas initial versions of the page-description language HTML only allowed a raw display of text, with inadequate formatting options, it is now possible to create arbitrary divisions (also called layers) to isolate and render any piece of content. Divisions are rectangular portions of the browser's screen with any defined location, size, background color, frame, and stacking depth. Divisions can contain other divisions, or overlap each other, in a stack where the highest hides the lower one. Technically, scripts can access properties of divisions, which mean that their arrangement might be dynamically modified, and consequently manipulated on user request by appropriate scripts.

When programmed appropriately, divisions can be used to render independent, overlaying windows, each displaying their own content. The extension of the HTML language towards divisions is called Dynamic HTML, or DHTML. The features of DHTML will also be used to implement the user interface according to the invention.

When enhanced with scripts and divisions, HTML can render very nice-looking pages of information, but it still lacks the functionality required for interactive applications. For instance, scripts would not suit the manipulation of images, or would not be powerful enough to navigate complex data warehouses. Such applications require real programming.

Applets, literally "small applications" are pieces of software intended to run within the user's browser. Like scripts and regular documents, applets are downloaded from the server but unlike scripts, they can further request and exchange additional information with that server.

Applets provide the appropriate framework for programming the actual window's behavior. As in any windowing system, some windows list documents, some provide application interfaces, etc. In the invention described, all window internal behavior will be performed by applets.

The user interface of the present invention is designed as client-side software that is automatically downloaded and executed on the browser, in the context of a Web-based client-server application. The software implements the components of a windowing system, like on regular workstations, but in the context of a Web page downloaded from a Web site.

The user interface object of the present invention relies on specific browser technologies that have been briefly described here above and is designed around two essential components:

A first component called the desktop, where individual windows are displayed. The desktop is implemented as a single Web page; a major part of it is the window manager that is a piece of DHTML code and JavaScript programming.

The second component is a set of several independent windows of various types, called hereafter WebWindows. The aspect and behavior regarding location, size, opening and closing, is commonly shared by all WebWindows; only their internal content differs. WebWindows are implemented as small Java applications (Applets).

At the first access, the browser will present a single page to the user that is downloaded from the web server like any other HTML page. This single page will be referenced hereafter as the desktop that may be seen as a surface where icons and windows will be laid down.

Figure 1 show a schematic representation of a desktop according to the present invention which comprises a title bar 1 for displaying label information and a working area 2 where individual WebWindows 3 are displayed.

Within the title bar 1 buttons or menus 4 are used to trigger actions to the system when the user click on them. For instance, these buttons or menus 4 may be used for the creation or the deletion of a WebWindow 3 or other desktop related operations

On the right side of the figure, within the title bar 1, icons 5 are displayed. These icons 5 are small glyphs that represent minimized objects like WebWindows 3 for example. Preferably, the icons 5 use different highlighting styles to reflect the underlying window status.

Within the working area 2 of the desktop, several WebWindows 3, are displayed. These WebWindows 3 are used for displaying information or for executing web enabled applications. Each WebWindow 3 also comprises a title bar 7 used for moving the WebWindow within the work area of the desktop and buttons 8 within the title bar for performing basic functions like for example closing, minimizing and maximizing the WebWindow 3.

The borders of a WebWindow 3 allows its resizing. Preferably, the arrows 6 used for resizing the WebWindows 3 are displayed on the screen only when the mouse cursor is over a the WebWindow frame.

Within the working area 2 of the desktop, individual WebWindows 3 reside, independently of others. Individual WebWindows 3 may also exchange information through the desktop. The desktop manages the graphical rendering of the WebWindows 3 thanks to a windows manager, but not their content. Individual WebWindows 3 display their interactive content transparently from the windowing system. The desktop provides an encapsulation mechanism that allows WebWindows 3 to behave like native Java applets.

The following window management operations are performed by the window manager that is incorporated in the desktop:

- Creation of a new WebWindow 3 of a given type with optional initial parameters.
- Positioning and optionally resizing of any WebWindow 3.
- Maximizing, minimizing a WebWindow. When a WebWindow 3 is minimized, it is converted to an icon 5 which is displayed in the title bar 1 of the desktop.
- Bringing a web window to the front and giving it user input focus, so that the user may interact with it by means of a pointing device or through the keyboard of the computer.
- Closing any WebWindow 3.
- Saving WebWindows state for next session.
- Restoring WebWindows state at beginning of session.
- Displaying short information messages or highlighting icons.

These basic operations are enumerated by way of example; obviously, other operations may be coded in the desktop manager part of the desktop.

Figure 2 represents schematically the software components downloaded and executed within the browser, and their data storage.

Within the DHTML side (on the right of the figure), the desktop page 9 is downloaded from the Web server 20, like any regular Web page. The desktop page 9 contains graphics, but mostly JavaScript methods 10, and their related storage 11. When JavaScript methods 10 are executed, windows divisions 12 are created, modified, or removed from the Document Object Model 13.

The Java side (on the left side of the figure) resides within the virtual machine or Java Plug-In 14 of the browser, and executes all applets 15,17. It also downloads and stores support Java libraries of classes. Individual WebWindows applets 17 store their necessary objects, and the JSObject 18 required for communicating with the DHTML side.

The Java and the DHTML parts communicate in both directions, using two mechanisms:

- Java Method Invocation from the DHTML side to any Applet's method.
- "JSObject" calls from within an applet 17 to the JavaScript methods 10 stored within the desktop page 9.

The "JSObject" class uses the Plug-In internal methods to call the surrounding browser's functionality.

Communication between both sides is bi-directional: JavaScript 10 has access to applets 17 through the document object model (DOM): it can call and pass arguments to as well as receive results from any applet's method.

WebWindows Applets 17 need the JSObject 18 in order to call JavaScript methods 10, also passing arguments and receiving results.

Communication with the Web server 20 uses the HTTP protocol, in two steps. First, the DHTML is downloaded as a Web page, and executed, and then individual applets 17 communicate directly with the server using any protocol.

The particularity of the architecture proposed here permits the graphical synchronization between DHTML components, and Java Applets: both are used dynamically, so that all objects can be created or removed during a session, as well as their properties modified.

Once the desktop is downloaded from the server as a single DHTML page. The page contains the initial, empty desktop presentation, the necessary JavaScript programs to operate all the Window Manager functions. Moreover, it downloads the necessary icon images for displaying the buttons 4,5.

The execution of JavaScript embedded within the desktop page results in the production of the desktop surface, and the setting up of the underlying software state to allow windows divisions and windows applets to be dynamically created, manipulated, and destroyed.

On the other hand, all applets are executed within a standard Virtual Machine that is plugged into the browser. The Java Plug-In provides the framework for running several applets, and will dynamically download them, as well as all support Java classes, from the Web or application server 20.

Both the desktop and each individual WebWindow 3 communicate with a Web server using the Web HTTP protocol. The desktop communicate with the server for downloading images, content or applets. The WebWindows 3 can use any protocol to communicate with the web server. In addition, WebWindows 3 communicate with the desktop for all window management operations that affect their appearance.

In order to implement powerful WebWindows, the capabilities of DHTML divisions and Java applets are combined; the overlaying of the various objects is controlled by the desktop.

Figure 3 shows schematically the overlaying of Java and DHTML objects within the browser. Each time a WebWindow is created, a DHTML division 21 is added dynamically to the document object model and its internal content is set to contain only one instance of the requested WebWindow applet.

Since the applet is a derived instance of the WinContainer class (see below), it instantiates a Java "WinContainer" panel 22 that displays in the created division. This implements the basic window functionality for overlaying windows (raise above, or lower below other windows).

The WinContainer installs appropriate graphical objects to render the window caption, title, buttons, borders, and their interactive behavior. This implements the functionality for minimizing, maximizing, resizing, and closing the window.

Finally, the WebWindow class creates its own panel 23, with only application specific widgets like lists, combo boxes, labels, etc. The panel 23 is then added to the WinContainer panel 22.

Optionally, when HTML or native documents must be displayed, another DHTML division 24 is overlaid on top of the applet, to show the document content. Notice that figure 3 emphasizes the stacking of the various graphical components. Actually, they are exactly superimposed so there is no visual effect other than a regular, simple window.

It is required that the Window Manager keeps the DHTML divisions and the applets in synchrony, matching their position, size, and overlay.

In order to simplify software implementation of the WebWindows, the synchronization part is commonly shared; therefore, the feature of class inheritance is used, as explained in the next section.

A WebWindow, implemented in Java, takes advantage from the concept of class inheritance in Object-Oriented languages. In the Java "Swing" GUI, all applets must derive from the JApplet class. Moreover, in the windowing system according to the present invention, all WebWindows applets derive from a common parent, called WinContainer, itself derived from JApplet. This class tree structure is depicted in figure 4.

With reference to figure 4, a standard component of the Swing graphical user interface is the JApplet, which implements a browser-resident application that interacts with the user through a rectangular area within the browser surface called JPanel.

By deriving WinContainer from JApplet, functionality for rendering an aspect of window (caption, title, buttons, and borders), and for enabling the user to interact with the window (dragging, raising, lowering, minimizing, maximizing, resizing, closing) is added. To that extent, the appropriate graphical widgets are added to the applet's panel. Finally, a JSObject is added and initialized for communication with the Window Manager by calling the desktop's JavaScript methods.

Lastly, real application logic is programmed in classes derived from WinContainer, which implement the actual GUI functionality and front-end processing of the WebWindows. This is done by creating a dedicated instance of JPanel, adding the required widgets, and embedding it within the ancestor's JPanel.

On the other hand, the Window Manager stores the current number and types of all open windows. All window-related information like the location, the size

or the depth of the window are stored in arrays. The window manager's JavaScript methods are capable to retrieve any applet's properties, and call any of their public methods. In this model, it only calls methods from WinContainer.

As for the Data Structures, specific algorithms are present in the WinContainer applet, and in the Window Manager. The way they interact is critical for implementing the windowing system.

The WinContainer applet defines notification methods for the following user input events

- Click on the minimize, maximize, and close button. The window is either minimized (hidden, leaving an icon), maximized to full screen, or closed (destroyed)
- Click on the window caption. The window obtains input focus; its depth is raised on top of the topmost window currently displayed. This causes the window to be completely displayed, possibly overlaying others. The JPanel of the derived WebWindow class is receiving input focus, and the window caption is highlighted.
- Click and drag (move with button pressed) on the window caption. The window is moved over the desktop.
- Click and drag on the window border. The window is resized by moving the border considered.

When one of the above events is detected:

- The WinContainer calls the required internal methods first (on close and focus events), and then calls the respective method in the Window Manager, via the JSObject. On drag events, mouse cursor coordinates are read from the event, and passed to the Window Manager.
- The Window Manager updates the window's DHTML divisions, by altering their parameters (hidden/visible, location, size, depth). When this is done, the desktop is automatically refreshed, and all divisions are

repainted. This automatically asks applets contained in the divisions to redraw their content, possibly adjusting to their new size.

On the other hand, user input events can also originate from the desktop: when user clicks on the buttons 2 for creating a new window, or when clicking on existing window's icons 5 in order to toggle their minimized/normal state.

In such a case, the JavaScript methods are directly called, which in turn asks applets to redraw their content.

The annexed source code includes a possible implementation of the two major components of the Windowing System of WebWindows. The Windowing System of WebWindows is based on a specific and particular interaction between two components, indicated in Figure 2:

- The Desktop, a DHTML and JavaScript program that is executed within the Browser. (see enclosed source code)
- Java Applets, or more specifically their parent class "WinContainer" (see Figure 4), that dialogs with the Desktop. (source code also included)

With reference to the enclosed source code, the following Desktop Data structures are defined.

Individual windows are created by loading an applet into a newly created DHTML division. The division is dynamically created and stored into the Browser's DOM. Moreover, a JavaScript array is used to keep track of references to the DOM divisions, in order to manage the input focus (which window receives keyboard input).

Buttons for creating new windows are defined in the DHTML page of the Desktop, hence they are also stored and accessed from the DOM. Actions are triggered by the Browser when buttons are pressed.

```
<a href="#"  onfocus="blur();" onmouseover="xxx_on.gif;" onmcuseout ="xxx_off.gif;"
onmousedown="xxx.src=xxx_down.gif;" onmouseup="xxx.src=xxx_on.gif;"
onclick="window_creation_script()">
```

```
<img name="xxx"  border=0  src="xxx_off.gif">
</a>
```

A special area of the Desktop is dedicated to representing the status of individual WebWindows (see Figure 1): a small changing icon represents minimized or maximized WebWindows. Actions on the icons, such as click, double-click, or rollover can change the display of the Desktop or the status of the WebWindow concerned. Small icons are implemented by loading a button (as above) into a small DHTML division.

The following data structures are stored in the WinContainer Applet.

The WinContainer stores the following data structures:

- A title bar with embedded buttons
- Four panels (north, south, east, west) that render the window border area
- A central panel (JPanel) in which application-specific content can be displayed, this is usually done by derived classes
- A JSObject to send invocation requests from the Applet to the DHTML JavaScript methods

All processing, either on the Desktop or on the Java Applets side, is triggered by user input actions. Such user actions are dispatched either to the Desktop (e.g. pressing a button) or the target Applet (e.g. clicking within the Applet panel, or typing some text). The following actions are monitored:

On the desktop (buttons and icons are configured to call handlers when the action occurs):

- Button pressed
- Small icon clicked or double-clicked

On WebWindows Applets (Java Listeners are implemented to be called when the action occurs) :

- Move
- Resize / Minimize / Maximize

- Raise / Lower
- Close
- Obtain / Loose input focus
- Keyboard

In most cases of user action, a particular listener detects the action; it creates a JavaScript method invocation with the appropriate parameters. The method to call is passed as a String, and the arguments as an array of Strings. The JSObject is used to call the Browser's JavaScript method. The result (of type Object) is typecasted to the appropriate data type, and that value is checked.

For some particular user actions (like loosing input focus because another window is gaining input focus) the event originates in the Desktop. To handle that, specific methods are implemented by the WinContainer object, and they can be called anytime from the Desktop's JavaScript. Such methods operate like listeners.

In both cases of user action, the WinContainer object will update its internal state and possibly its graphical layout (e.g. the color of the title bar depending of the input focus).

The following algorithms are implemented in the Desktop.

When the user action is targeting the desktop (e.g. a button is pressed), a specific JavaScript handler is called, which does the necessary operation to the Desktop and possibly calls existing WebWindow's methods to handle the action. For instance, for creating a new WebWindow, the following fragment of pseudo-code is executed:

```
// Create new WebWindow container (a DHTML division). This method is called when a
// button is pressed on the Desktop
var container = document.createElement("<div>…</div>");   // Create DIVision
container.innerHTML = "<APPLET>…</APPLET>";               // Load applet into it
// Add reference to the new container in a JavaScript array
winList[windowIndex++] = container;

// Create the small icon in Desktop bar
var handle = document.desktopbar.createElement("<div>…</div>");  // Small DIVision
handle.innerHTML = "<A HREF=>…<IMG…></A>";                       // with a button inside
```

Otherwise, when the user action was targeting a WebWindow, the event was caught in the WinContainer Applet, and possibly dispatched to the Desktop by invoking a JavaScript method that does the necessary processing.

In both cases of user action, the Desktop will update its internal state and possibly its graphical layout according to the action.

By way of example, when the user clicks on the title bar of a lowered WebWindow, and moves the mouse around, the system behavior is as follows:

The WebWindow gains input focus, is raised on top of other windows, and is moved.

The following events and communications occur between the Desktop and the WebWindow

Step	Desktop	WebWindow
1		User action "click" detected. Listener for the title bar is called.
2		Listener changes title bar style (to show that window gained input focus)
3		Listener invokes Desktop method: activateWindow(this)
4	Method activateWindow(target): use internal variables and array winList to find the WebWindow that currently has input focus. Calls a method in that WebWindow to tell it to loose input focus.	
5	For the target WebWindow, update the Z property of the DHTML division – window is raised on top of others	
6		User action "move" detected. Listener for mouse motion called.
7		Motion increment (delta_x, delta_y) computed, and assembled into a parameter list.
8		Listener invokes Desktop method: moveWindow(this, dx, dy)

9	Method moveWindow(target, dx, dy): extraction of motion increment.
10	For the target WebWindow, update the location properties (X,Y) of the DHTML division – window moves on screen

For resizing WebWindows, the same principle applies, with different methods called and different parameters.

Hereunder is the HTML source code for a possible implementation of the Destktop component.

```
<HTML XMLNS:IE>
<HEAD>
<TITLE>The Knowledge Factory</TITLE>

<!--
/*****************************************************************
 (c) AKODI SA - The Knowledge Factory
 $RCSfile: akodium.html,v $
 $Revision: 1.42 $
 $Date: 2001/05/21 07:46:40 $
 $Name:  $
 $Author: burtin $
-->

<STYLE>
@media all {
   IE\:HOMEPAGE {behavior:url(#default#homepage)}
}
</STYLE>

<SCRIPT LANGUAGE="JavaScript">
//  address for plugin auto-download, dont forget to change it if
//  you are using a new pluging version
var sPluginAddr          = "http://java.sun.com/products/plugin/1.3/jinstall-13-
win32.cab#Version=1,3,0,0";
var JAVA_PLUGIN_CLASSID = "clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"

//-------------------------------------------------------------------------//

//---------------------------------------------------------------------------
Connection object
//
// used to retreiveserer Addresse & protocol.
//---------------------------------------------------------------------------

var myConnection     = null;   // the connection object

function ConnectionData() {
       this.sServerAddr     = document.location.host;
       var temp = document.location.protocol;
       this.sServerProtocol = temp.substr(0,temp.length-1);
       this.sUsername              = '';
       this.sUser                  = '';
       this.sCompany       = '';

}
```

```
//-----------------------------------------------------------------------
// current actived object
//-----------------------------------------------------------------------

var currentActivatedWindow = null;
var mustBeRelease = true;

//-----------------------------------------------------------------------// some
GLOBAL COSNTANTS
//-----------------------------------------------------------------------

var iMaxWindow       = 100;    // max of open windows
var ixx              = 100;    // default Window X position
var iyy              = 100;    // default Window Y position
var deskMgr          = null;   // the desktop manager;
var debugMgr         = null;   // debug manager
var restoreMode      = false;  // used for window restore; when this flag is
true, it means a restauration is running;
var frontZindex      = 1;      // used for computing new windows new indexes.
var iDebugLevel      = 0;      // the debug level (info are shown in the debug
window)
var STATUS_NOTCREATED= -1;     // somme constants for window statuts
var STATUS_PENDING   = 1;      // ..
var STATUS_LAUNCHED  = 2;      // ..
var STATUS_RUNNING   = 3;      // ..
var STATUS_TOBEKILLED= 4;      // ..
var aWinList         = new Array(iMaxWindow);  // array of WindowObjects , SEE
BELOW

//-----------------------------------------------------------------------//
Window object
//
//-----------------------------------------------------------------------

// iRestore      : its a  window restoring (given to applet as parameter)
// sApplet       : the applet name class file
// sCodebase     : the code base directory
// sImageDir     : the image direxTory (i'm not sure its usefull)
// iLeft,iTop    : position
// iDesk         : desktop
// iSizeX,iSizeY : size
// iIcoMax       : integer bit array: bit 0: iconized ; bit 1 :maximized ;
// aParamList    : optionnal array of parameters given to applet, example
[PARAMNAME0,PARAMVALUE0,PARAMNAME1,PARAMVALUE1,...]
// handleIcon    : andle icon filename
// bIsActiveX    : does it contains an activeX?
// iActiveXtop   : ActiveX Y position from top of window
// sDoc          : ActiveX URL
// sPkid         : given to applet (ask Stephane)


function
WindowObject(iRestore,sApplet,sCodebase,sImageDir,sArchive,sVersion,iLeft,iTop,iD
esk,iSizeX,iSizeY,iIcoMax,aParamList,sHandleIcon,bIsActiveX,iActiveXtop,sDoc,sPki
d) {
      // find an empty slot in   aWinList array for registering
      this.bDivStatus     =STATUS_NOTCREATED;
      var i=0;
      while((i<iMaxWindow) && (aWinList[i]!=null)){i++;}
      if(i>=iMaxWindow) {
            alert("too many windows open, please close some");
            return;
      }
      this.iMyIndex   = i;           // index in array of windows objects
```

```
        // init some internal data
        this.className      = sApplet; // not really usefull,but never know  )
        this.sWinName       = 'WindowFrame'+(i); // div name
        this.bDivMinimiz    = false;   // is div minimized ?
        this.blinking       = false;   // does the handle blink
        this.sDivHandle     = sHandleIcon;    // div handle icon name
        this.iDivDesk       = iDesk;          // desktop where the div is.
        this.bDivKObject    = bIsActiveX;     // true if div contains a KObject
(activeX)
        this.bKObjectVisible= false;          // is Kobject visible ?
        this.iDivKObjectTop = iActiveXtop;  // KObject top position
        this.bDivStatus     = STATUS_PENDING;  // true if the applet in the div is
running
        this.oWindowElement = null;   // the element created for the Applet DIV
        this.oAppletElement = null;   // the element created for the Applet
        this.oHandleElement = null;   // the element created for the Handle (icon)
        this.oFiletElement  = null;   // the element created for string between
window and handle
        this.oKobjElement   = null;   // the element created for string between
window and handle
        this.bHaveFocus     = true;   // does it have focus or not ? (internal,
use get/looseFocus())
        this.sHandleIcon    = sHandleIcon;  // handle image name
        this.sUniqWinID     = "";           // unique Window ID used to prohibit
opening same object x2.
        this.bCanBeClosed   = true;         // can it be closed ?

        // does the window fit in the desktop ?
        if ((sHandleIcon)=="") this.sHandleIcon="handle_default";
        if (parseInt(iSizeX)+parseInt(iLeft)> iGiveMeDeskTopWidth()) iLeft=10;
        if (parseInt(iSizeY)+parseInt(iTop)> iGiveMeDeskTopHeight()) iTop=60;

        var originalLeft   = iLeft;
        var originalTop    = iTop;
        var originalWidth  = iSizeX;
        var originalHeight = iSizeY;
        if ((iIcoMax & 2)==2)  //  maximized
      { iLeft=0;iTop=54; iSizeX= iGiveMeDeskTopWidth();
iSizeY=iGiveMeDeskTopHeight()-55;}

        // Window object methods initialisation, see each function def for details
        this.closeWindow    = WindowObject_closeWindow;
        this.activateWindow = WindowObject_activateWindow;
        this.getLeft        = WindowObject_getLeft;
        this.getTop         = WindowObject_getTop;
        this.getWidth       = WindowObject_getWidth;
        this.getHeight      = WindowObject_getHeight;
        this.moveIt         = WindowObject_moveIt;
        this.resizeIt       = WindowObject_resizeIt;
        this.showKObj       = WindowObject_showKObj;
        this.hideKObj       = WindowObject_hideKObj;
        this.resizeKObj     = WindowObject_resizeKObj;
        this.isRunning      = WindowObject_isRunning;
        this.hideWin        = WindowObject_hideWin;
        this.showWin        = WindowObject_showWin;
        this.hideHandle     = WindowObject_hideHandle;
        this.showHandle     = WindowObject_showHandle;
        this.hideFilet      = WindowObject_hideFilet;
        this.showFilet      = WindowObject_showFilet;
        this.setKObjvisible = WindowObject_setKObjvisible;
        this.launch         = WindowObject_launch;
        this.setRunning     = WindowObject_setRunning;
        this.bringToFront   = WindowObject_bringToFront;
        this.getFocus       = WindowObject_getFocus;
        this.releaseFocus   = WindowObject_releaseFocus;
```

```
        this.invertState    = WindowObject_invertState;
        this.getHandleImg   = WindowObject_getHandleImg;
        this.setHandleTip   = WindowObject_setHandleTip;
        this.deskTopChanged = WindowObject_deskTopChanged;
        this.hideAll        = WindowObject_hideAll;
        this.setNotRunning  = WindowObject_setNotRunning;
        this.moveToDesk     = WindowObject_moveToDesk;
        this.getZindex      = WindowObject_getZindex;
        this.setBlink       = WindowObject_setBlink;
        this.setClosable    = WindowObject_setClosable;

        // the Applet object
        var sContent= '<OBJECT classid="'+JAVA_PLUGIN_CLASSID+'"
NAME="'+this.sWinName+'APPLET" '+
                        'WIDTH = "100%" HEIGHT = "100%"  '+
                    'codebase="'+sPluginAddr+'"> '+
                    '<NOEMBED><XMP>'+
                    '</APPLET></XMP>'+
                    '<PARAM NAME = CODE        VALUE = "'+sApplet+'" > '+
                    '<PARAM NAME = CODEBASE    VALUE = "'+sCodebase+'" > '+
                    '<PARAM NAME = "type"      VALUE = "application/x-java-
applet;version1.3"> '+
                    '<PARAM NAME = IMAGEDIR    VALUE = "'+sImageDir+'" > '+
                    '<PARAM NAME = ARCHIVE     VALUE = "'+sArchive+'" > '+
                    //'<PARAM NAME = CACHE_OPTION     VALUE = "Plugin" > '+
                    //'<PARAM NAME = CACHE_ARCHIVE    VALUE = "'+sArchive+'" > '+
                    //'<PARAM NAME = CACHE_VERSION    VALUE = "'+sVersion+'" > '+
                        '<PARAM NAME = "winname"      VALUE = "'+this.sWinName+'">
'+
                          '<PARAM NAME = "scriptable"    VALUE = "true">'+     //
VERY IMPORTANT, communications between HTML and Java
                          '<PARAM NAME = "MAYSCRIPT"      VALUE = "true">'+  //
wont work without these 2 lines
                          '<PARAM NAME = "APPLETWIDTH"    VALUE =
"'+originalWidth+'">'+
                          '<PARAM NAME = "APPLETHEIGHT"   VALUE =
"'+originalHeight+'">'+
                          '<PARAM NAME = "APPLETLEFT"     VALUE =
"'+originalLeft+'">'+
                          '<PARAM NAME = "APPLETTOP"      VALUE =
"'+originalTop+'">'+
                            '<PARAM NAME = "APPLETSTATE"    VALUE =
"'+iIcoMax+'">'+
                          '<PARAM NAME = "DESKTOPWIDTH"   VALUE =
"'+iGiveMeDeskTopWidth()+'">'+
                    '<PARAM NAME = "DESKTOPWIDTH"   VALUE = "'+iDesk+'">'+
                          '<PARAM NAME = "DESKTOPHEIGHT"  VALUE =
"'+iGiveMeDeskTopHeight()+'">'+
                          '<PARAM NAME = "serverAddr"     VALUE =
"'+myConnection.sServerAddr+'">'+
                          '<PARAM NAME = "serverProtocol" VALUE =
"'+myConnection.sServerProtocol+'">'+
                    '<PARAM NAME = "company"        VALUE =
"'+myConnection.sCompany+'"> '+
                    '<PARAM NAME = "RESTORESTATE"   VALUE = "'+iRestore+'">'+
                    '<PARAM NAME = "DEBUGLEVEL"  VALUE = "'+iDebugLevel+'">';



        // some addtitionnal and optional parameters

        if (aParamList!=null)
              for (var i=0;i<(aParamList.length/2);i++)
           {  sContent=sContent+'<PARAM NAME    ="'+aParamList[2*i]+'" VALUE =
"'+aParamList[2*i+1]+'">';}
```

```
   if (sPkid!=null)      sContent=sContent+'<PARAM NAME    = "pkid"        VALUE =
"'+sPkid+'"> ';
   if (sDoc!=null)       sContent=sContent+'<PARAM NAME    = "URL"         VALUE =
"'+sDoc+'"> ';

   sContent=sContent+'</APPLET></NOEMBED></OBJECT>';

   // some addtionnal content for optionnal IE activeX
   if (bIsActiveX)
    { var w=iSizeX-20;
        var h=iSizeY-iActiveXtop-10;
     // IE ActiveX
        sContent=sContent+'<object classid="clsid:8856F961-340A-11D0-A96B-
00C04FD705A2" id="'+this.sWinName+'KOBJECT"
style="position:absolute;visibility:hidden; top:'+iActiveXtop+'; left:10;
width:\''+w+'\'; height:\''+h+'\';">'+
        '<param name="ExtentX"                value="13282">'+
        '<param name="ExtentY"                value="11324">'+
        '<param name="ViewMode"               value="0">'+
        '<param name="Offline"                value="0">'+
        '<param name="Silent"                 value="0">'+
        '<param name="RegisterAsBrowser"      value="0">'+
        '<param name="RegisterAsDropTarget"   value="0">'+
        '<param name="AutoArrange"            value="0">'+
        '<param name="NoClientEdge"           value="0">'+
        '<param name="AlignLeft"              value="0">'+
        '<param name="NoWebView"              value="0">'+
        '<param name="HideFileNames"          value="0">'+
        '<param name="SingleClick"            value="0">'+
        '<param name="SingleSelection"        value="0">'+
        '<param name="NoFolders"              value="0">'+
        '<param name="Transparent"            value="{0057D0E0-3573-11CF-AE69-
        '<param name="ViewID"                   value="'+sDoc+'">'+
08002B2E1262}">'+
        //'<param name="Location"
        '</object>';
       }
    this.data = sContent;
   //  the CONTENT CREATION
   //
   var iZindex = frontZindex++;


    Log(sApplet+': desk='+iDesk+' state= '+iIcoMax+'
currentdesk='+deskMgr.currentDesktop);

   // then win create the 4 element for the window
   //  handledata the little image attach to the window on top of desktop
   //  windiv the window (i.e. tha applet) DIV
   //  handlediv, the div which contain the handle
   // filetdiv, the div with the dashed line betwen handle and window

   // compute elements visibility
   var hVisible  = 'hidden';
   var fVisible  = 'hidden';
   var wVisible  = 'visible';
   if ((deskMgr.currentDesktop==iDesk)||(iDesk==0)||(iDesk==-1))
    { hVisible='visible';
      if ((iIcoMax & 1)==0)  {fVisible='visible'; wVisible='visible';}
    }
   if ((iIcoMax & 1)==1) this.bDivMinimiz=true;
   Log(sApplet+': hVisible='+hVisible+' fVisible= '+fVisible);
```

```
  // the element shape is different depending of the window state
  // if the handle icon filename given in parameter is XXXX
  // we append '_c.gif' if the window have a regular state
  // we append '_r.gif"  if the window is minimized
  var handleShape="c"; if (this.bDivMinimiz) handleShape="r";

  this.handledata= '<a href="#" onfocus="blur();" '+
                   'onclick   ="aWinList['+this.iMyIndex+'].invertState();" '+
                   //'onmouseover
="aWinList['+this.iMyIndex+'].activateWindow();" '+

'ondblclick="aWinList['+this.iMyIndex+'].closeWindow();">'+
                   '<img name="'+this.sWinName+'HANDLE"  border=0
src="pictures/desktop/'+this.sHandleIcon+'_'+handleShape+'.gif" alt="">'+
                          '</a>';
  this.windiv    = '<div id="'+this.sWinName+'" style="'+

                          'position: absolute; visibility:'+wVisible+';'+
                          'z-index: '+iZindex+'; '+
                          'left: '+iLeft+'px; top: '+iTop+'px; '+
                        'width: '+iSizeX+'px; height: '+iSizeY+'px">'+

                          '</div>';
  this.handlediv = '<div id="'+this.sWinName+'H" style="'+

                          'position: absolute;  visibility:'+hVisible+';'+
                          'z-index:'+(iZindex)+'; left: '+iLeft+'px; top: 33px;
width: 20px; height: 20px">'+
                          '</div>';
  this.filetdiv = '<div id="'+this.sWinName+'F" style="'+

                          'position: absolute;  visibility:'+fVisible+';'+
                            'background-image:
url(pictures/desktop/slashdots.gif); '+
                          'z-index:0; left: '+iLeft+'px; top: 34px; width: 20px;
height: '+iTop+'px">'+
                          '</div>';

  ixx = 30 + originalLeft; // next window position
  iyy = 30 + originalTop;
  aWinList[this.iMyIndex]=this;  // registering
  this.launch();
 }

// change a window desk, if desk is -1 is means all desks
function WindowObject_moveToDesk(iDesk)
 { this.iDivDesk=iDesk;
   if (this.iDivDesk!=deskMgr.currentDesktop) this.deskTopChanged(iDesk);
 }

//
// INTERNAL FUNCTIONS

function WindowObject_launch() {
      this.bDivStatus    =  STATUS_LAUNCHED;
    this.oFiletElement=document.createElement(this.filetdiv);
    document.body.appendChild( this.oFiletElement);

    this.oHandleElement=document.createElement(this.handlediv);
    document.body.appendChild( this.oHandleElement);
      this.oHandleElement.innerHTML=this.handledata;

    this.oWindowElement=document.createElement(this.windiv);
    document.body.appendChild( this.oWindowElement);
    this.oWindowElement.innerHTML=this.data;
```

```
    this.oAppletElement=document.applets(this.sWinName+'APPLET');
    this.oKobjElement=document.all(this.sWinName+'KOBJECT');

      this.oWindowElement.style.clip = "rect(0 auto auto 0)";
}

function WindowObject_setClosable(bClosable)
  {this.bCanBeClosed=bClosable;}

// guest what:  this close the window
function WindowObject_closeWindow() {
      if (!this.bCanBeClosed) return ; // the window cant be closed, request is
ignored !

      if (this.isRunning()) {// call the doOnClosing() method in the java applet
            this.oAppletElement.doOnClosing();
      }

      if (iDebugLevel>0) Log('Destroying '+this.sWinName);

      document.body.removeChild( this.oFiletElement);
      document.body.removeChild( this.oHandleElement);
      // warning: if we remove the applet child node,  random crashes may appear
      this.oWindowElement.removeChild( this.oAppletElement);
      if(this.oKobjElement != null)
            this.oWindowElement.removeChild( this.oKobjElement);

      // remove the applet while overwriting  the div content
      this.oWindowElement.innerHTML="";
      document.body.removeChild( this.oWindowElement);

      // it may help the garbage collector
      this.oHandleElement=null;
      this.oFiletElement=null;
      this.oWindowElement=null;
      this.oKobjElement=null;
      this.oAppletElement=null;
      aWinList[this.iMyIndex]=null; // unregister
      currentActivatedWindow=null;
      giveFocusToFrontWindow();     // find a new window and give it focus.
}

// set the window  handle  tool tip
function WindowObject_setHandleTip(sLabel)
 { this.getHandleImg().alt=sLabel; }

// set the window status to RUNNING
function WindowObject_setRunning() {
      this.bDivStatus=STATUS_RUNNING;
      this.getFocus();
}

// give the focusd to the window

function WindowObject_getFocus() {

      this.bHaveFocus=true;
      if(currentActivatedWindow != null) currentActivatedWindow.releaseFocus();
      currentActivatedWindow = this;
}

// release the focus

function WindowObject_releaseFocus()
```

```
 { if (this.bHaveFocus)
   { this.bHaveFocus=false;
     if (this.isRunning()) {
        this.oAppletElement.setSelected(false);
     }
   }
 }

// tells the windows is not running anymore (I'm not sure its called anymay)
function WindowObject_setNotRunning()
 {this.bDivStatus=STATUS_TOBEKILLED;}

// says if the window (the applet) is currently running
function WindowObject_isRunning()
 { return (this.bDivStatus==STATUS_RUNNING); }

// called we the current desktop changes
function WindowObject_deskTopChanged(n)
 { if (this.iDivDesk >0)
   { if  (this.iDivDesk==n)
        { this.showHandle();
          if (!this.bDivMinimiz)
             { this.showWin();
               this.showFilet();
               if (this.bKObjectVisible) this.showKObj();
             }
        }
       else
        { this.hideWin();
          this.hideHandle();
             this.hideFilet();
             if (this.bKObjectVisible) this.hideKObj();
        }
   }
 }

// SIZE & POSITIONNING FUNCTION
function   WindowObject_getLeft()        {return
parseInt(this.oWindowElement.style.left);}
function   WindowObject_getTop()         {return
parseInt(this.oWindowElement.style.top);}
function   WindowObject_getWidth()       {return
parseInt(this.oWindowElement.style.width);}
function   WindowObject_getHeight()      {return
parseInt(this.oWindowElement.style.height);}

function   WindowObject_resizeIt(iX, iY, iW , iH) {
  this.moveIt(iX,iY);
  this.oWindowElement.style.width=iW;
  this.oWindowElement.style.height=iH;
  if (this.bDivKObject) this.resizeKObj(iW-20,iH-10-this.iDivKObjectTop);
}


function   WindowObject_moveIt(iX,iY) {

    if (iX>iGiveMeDeskTopClientWidth())  iX = iGiveMeDeskTopClientWidth() -20;

    if (iY>iGiveMeDeskTopClientHeight()) iY = GiveMeDeskTopClientHeight()-20;

    if (iX<0) iX=0;
    if (iY<55) iY=55;
    this.oWindowElement.style.left=iX;
    this.oWindowElement.style.top=iY;
```

```
    this.oHandleElement.style.left=iX;

    this.oFiletElement.style.left=iX;
    this.oFiletElement.style.height=iY;
}

//kobject function
function   WindowObject_showKObj()       {if (this.bDivKObject)
this.oKobjElement.style.visibility ='visible';}
function   WindowObject_hideKObj()       {if (this.bDivKObject)
this.oKobjElement.style.visibility ='hidden';}
function   WindowObject_hideWin()
{this.oWindowElement.style.visibility='hidden';}
function   WindowObject_showWin()
{this.oWindowElement.style.visibility='visible';}
function   WindowObject_hideHandle()
{this.oHandleElement.style.visibility='hidden';}
function   WindowObject_showHandle()
{this.oHandleElement.style.visibility='visible';}
function   WindowObject_hideFilet()
{this.oFiletElement.style.visibility='hidden';}
function   WindowObject_showFilet()
{this.oFiletElement.style.visibility='visible';}
function   WindowObject_getHandleImg()   {return
this.oHandleElement.all(this.sWinName+'HANDLE');}
function   WindowObject_hideAll()
{this.hideHandle();this.hideFilet();this.hideKObj();this.hideWin();}
function
WindowObject_resizeKObj(iW,iH){this.oKobjElement.style.width=iW;this.oKobjElement
.style.height=iH;}
function   WindowObject_setKObjvisible(bState) {
  this.bKObjectVisible=bState;
  if(deskMgr.currentDesktop==this.iDivDesk) {
    if (bState)
      this.showKObj();
    else
      this.hideKObj();
   }else {
     this.hideWin();
    }
  }

  // sets a window state iconized -> nomal -> iconized
  function   WindowObject_invertState()
  {
    this.bDivMinimiz=!this.bDivMinimiz;
    if (this.bDivMinimiz)
     {this.hideWin();this.hideFilet();
      if (this.bKObjectVisible) this.hideKObj();
        if  (this.Blinking)
    this.getHandleImg().src='pictures/desktop/'+this.sHandleIcon+'_r_blink.gif';
                  else
    this.getHandleImg().src='pictures/desktop/'+this.sHandleIcon+'_r.gif';
       giveFocusToFrontWindow();
     }
    else
     {this.showWin();this.showFilet();
      if (this.bKObjectVisible) this.showKObj();
        if  (this.Blinking)
    this.getHandleImg().src='pictures/desktop/'+this.sHandleIcon+'_c_blink.gif';
                  else
    this.getHandleImg().src='pictures/desktop/'+this.sHandleIcon+'_c.gif';
     this.activateWindow();
     this.oAppletElement.setSelectedByHtml();
      }
```

```
}

// make the handle blink or not
// iState=0 : off else : on
//
// images files  named HandleIcon_r_blink.gif & HandleIcon_c_blink.gif
// must exists in pictures/desktop/ dir

function   WindowObject_setBlink(iState)
 { if (iState==0) this.Blinking=false; else this.Blinking=true;
   if  (this.Blinking)
    { if (this.bDivMinimiz)
this.getHandleImg().src='pictures/desktop/'+this.sHandleIcon+'_r_blink.gif';
        else
this.getHandleImg().src='pictures/desktop/'+this.sHandleIcon+'_c_blink.gif';
      }
  else
    { if (this.bDivMinimiz)
this.getHandleImg().src='pictures/desktop/'+this.sHandleIcon+'_r.gif';
        else
this.getHandleImg().src='pictures/desktop/'+this.sHandleIcon+'_c.gif';
      }
 }

// return the window  Z index
function WindowObject_getZindex()
 { return  this.oWindowElement.style.zIndex;}

// brings the window to front
function   WindowObject_bringToFront() {
   this.oWindowElement.style.zIndex=frontZindex++;
   this.oHandleElement.style.zIndex=frontZindex;
}

// activates the window
function WindowObject_activateWindow() {

  this.getFocus();

  this.bringToFront();

}


// return the top window
function getWindowOnFront()
 { var zmin=-1;
   var n=-1;
   for (var i=0;i<iMaxWindow;i++)
      if (aWinList[i]!=null)
          if (!aWinList[i].bDivMinimiz)
            if (parseInt(aWinList[i].oWindowElement.style.zIndex)>parseInt(zmin))
{zmin=aWinList[i].oWindowElement.style.zIndex;n=i;}
    if (n>=0) return aWinList[n]; else return null;
 }

// give the focus to the top window
function  giveFocusToFrontWindow() {
      var w = getWindowOnFront();
      if (w!=null) {
            if(!w.bHaveFocus) {
                  if(w.isRunning()) {
                        w.oAppletElement.setSelectedByHtml();
                        w.getFocus();
                  }
```

```
            }
        }
    }
// just say if a we can open a window using sUniqWinID key.
function canIOpenWindow(sUniqID) {
  Log('Try to open '+sUniqID);
  if (sUniqID=="") return true;
  for (var i=0;i<iMaxWindow;i++) {
    if (aWinList[i]!=null) {
      if(aWinList[i].bDivStatus<STATUS_TOBEKILLED) {
                    if  (aWinList[i].sUniqWinID==sUniqID) {
          if(aWinList[i].iDivDesk > 0)deskMgr.selectDesk(aWinList[i].iDivDesk);
          if (aWinList[i].bDivMinimiz) {
            aWinList[i].invertState();
          }else {
            aWinList[i].activateWindow();
          }
                        return false;
        }
      }
    }
  }
  return true;
}

// RETREIVE a windows object according to its logical name
// lagical names are "WindowFrameX" where X is an integer
function getWindowObject(it) {
      var index = it.substr(11);
      return aWinList[parseInt(index)];
}

// debug purpose
//
function cfj(it,fct)   { if (iDebugLevel>1) Log('Call from java
('+it+':'+fct+')');}
function Log(sMsg)                          {debugMgr.logMsg(sMsg);}
function stopRestoreMode()                  {restoreMode=false;
SetButtonsPanel();ixx=30;iyy=100; Log('-->'+ixx+' '+iyy);if (iDebugLevel>0)
Log('restoreMode OFF');}

// functions called from Java Applets.
function closeObject(it)
{cfj(it,'closeObject');getWindowObject(it).setNotRunning();getWindowObject(it).hi
deAll(); setTimeout('getWindowObject("'+it+'").closeWindow();',1000);}
function moveObject( it, iX , iY)
{cfj(it,'moveObject');getWindowObject(it).moveIt(iX,iY);}
function setHandleTip (it,label)
{cfj(it,'setHandleTip');getWindowObject(it).setHandleTip(label);   }
function resizeObject(it,iX,iY,iW,iH)
{cfj(it,'resizeObject');getWindowObject(it).resizeIt(iX,iY,iW,iH);}
function maximizeObject(it)
{cfj(it,'maximizeObject');getWindowObject(it).resizeIt(0,0,iGiveMeDeskTopWidth(),
iGiveMeDeskTopHeight());}
function showKObject(it)
{cfj(it,'howKObject');getWindowObject(it).setKObjvisible(true);}
function hideKObject(it)
{cfj(it,'hideKObject');getWindowObject(it).setKObjvisible(false);}
function iObjectLeft(it)                    {cfj(it,'iObjectLeft');return
getWindowObject(it).getLeft();}
function iObjectTop(it)                     {cfj(it,'iObjectTop');return
getWindowObject(it).getTop();}
```

```
function invertState(it)
{cfj(it,'invertState');getWindowObject(it).invertState();}
function setBlink(it,iState)              {cfj(it,'setBlink:
'+iState);getWindowObject(it).setBlink(iState);}
function setUser(usrname,user,company)
{myConnection.sUsername=usrname;myConnection.sUser=user;myConnection.sCompany=com
pany;setTimeout("removeSplashScreen();",5000);SetTimeAndUser();}
function getUser(user)                    {return myConnection.sUser;}
function startRestoreMode()               {restoreMode=true;if (iDebugLevel>0)
Log('restoreMode ON');}
function setClosable(it,bClosable)        {cfj(it,'setClosable:
'+bClosable);getWindowObject(it).setClosable(bClosable==1);}
function callHelpSystem(sFilename,sAnchor){cfj('?','callingHelp
'+sFilename+'#'+sAnchor); callHelp(sFilename,sAnchor);}
function setDocLocation(it,id)            {cfj(it,'setDocLocation');
document.all(it+"KOBJECT").navigate(
myConnection.sServerProtocol+"://"+myConnection.sServerAddr+"/servlet/kobj?cmd=do
wnload&id="+id);}

function activateWindow(it)
{cfj(it,'activateWindow');getWindowObject(it).activateWindow();}
function amIRunningInDhtm()                              {return 1;} // this
function is called by applets containers to test their environnement

// windows restoring: these functions are called by WhoAmI java applet (the
splash screen)
function restoreTreeWin(ix,iy,iw,ih,ic,idesk)
       { if (iDebugLevel>0) Log('restoring a tree');
            var cmd =
'launchTeee(1,'+ix+','+iy+','+iw+','+ih+','+ic+','+idesk+');';
         setTimeout(cmd,50);
          }

function restoreRecomWin(ix,iy,iw,ih,ic,idesk)
       { if (iDebugLevel>0) Log('restoring a recommendation viewer');
            var cmd =
'launchRecommendations(1,'+ix+','+iy+','+iw+','+ih+','+ic+','+idesk+');';
         setTimeout(cmd,50);
          }

function restoreIagentWin(ix,iy,iw,ih,ic,idesk)
       { if (iDebugLevel>0) Log('restoring an Iagent');
            var cmd =
'launchIagent(1,'+ix+','+iy+','+iw+','+ih+','+ic+','+idesk+');';
         setTimeout(cmd,50);
          }

 function restoreKobjWin(sRid,ix,iy,iw,ih,ic,idesk)
       { if (iDebugLevel>0) Log('restoring an Kobj');
            var cmd =
'launchKobj(1,'+sRid+','+ix+','+iy+','+iw+','+ih+','+ic+','+idesk+');';

         setTimeout(cmd,50);
          }

function restoreWsWin(sId,kId,ix,iy,iw,ih,ic,idesk)
   {if (iDebugLevel>0) Log('restoring a Workspace');
   var cmd =
'launchWs(1,'+sId+',0,'+kId+','+ix+','+iy+','+iw+','+ih+','+ic+','+idesk+');';

   setTimeout(cmd,50);

  }

 //  very important:  each applet must call this function in the
```

```
// start() method, otherwise it wont work
function iAmRunning(it)
   {cfj(it,'iAmRunning');
    win = getWindowObject(it);
    win.setRunning();
      if
((win.bDivMinimiz)||((win.iDivDesk>0)&&(win.iDivDesk!=deskMgr.currentDesktop)))
win.hideWin();  // because applets dont start if invisible
      if (iDebugLevel>0) Log (it+" is running.");
    if (restoreMode) {setTimeout('var res=SPLASHAPPLET.launched(); if (res==1)
stopRestoreMode();',50); }
   }

// internal functions
function   iGiveMeDeskTopClientHeight() {return document.body.clientHeight;}
function   iGiveMeDeskTopClientWidth()  {return document.body.clientHeight;}
function   iGiveMeDeskTopHeight()       {return document.body.clientWidth;}
function   iGiveMeDeskTopWidth()        {return document.body.clientWidth;}

// ------------------------------------------------------------------------
// Splash screen fnction
// ------------------------------------------------------------------------
var splash=null;
function launchSplashScreen()
 { sContent= '<OBJECT classid="'+JAVA_PLUGIN_CLASSID+'" NAME="SPLASHAPPLET" '+
             'WIDTH = "100%" HEIGHT = "100%"  '+
             'codebase="'+sPluginAddr+'"> '+
             '<NOEMBED><XMP> '+
             '<APPLET> </XMP> '+
             '<PARAM NAME = CODE                    VALUE =
"akodi.client.whoami.whoami.class" > '+
             '<PARAM NAME = CODEBASE                VALUE = "applet" > '+
             '<PARAM NAME = "type"                  VALUE = "application/x-java-
applet;version1.3"> '+
             '<PARAM NAME = IMAGEDIR                VALUE = "pictures/whoami" > '+
             '<PARAM NAME = ARCHIVE                 VALUE =
"whoami.jar,interface.jar,akodi.jar" > '+
             '<PARAM NAME = "scriptable"            VALUE = "true">'+ // VERY
IMPORTANT, communicatins between HTML and Java
             '<PARAM NAME = "MAYSCRIPT"             VALUE = "true">'+ // wont
work without these 2 lines
             '<PARAM NAME = "serverAddr"            VALUE =
"'+myConnection.sServerAddr+'">'+
             '<PARAM NAME = "serverProtocol" VALUE =
"'+myConnection.sServerProtocol+'">'+
            '</APPLET></NOEMBED></OBJECT>';
  var div    ='<div id="SPLASH" style="'+
              'position: absolute; '+
              'z-index: 10000; '+
              'left: '+((iGiveMeDeskTopWidth()-500)/2)+'px;
top:'+((iGiveMeDeskTopHeight()-100)/2)+'px; '+
             'width: 500px; height: 100px">'+

              '</div>';
   splash=document.createElement(div);
   document.body.appendChild(splash);
   splash.innerHTML=sContent;
  }

function removeSplashScreen()
  {splash.style.visibility='hidden';
   // we only hide it, beacause the splash screen is still usefull, the
   // whoami applet can be called from time to time. (save layout feature).
  }
```

```
// ---------------------------------------------------------------------------
// System INIT, called when page loaded
//----------------------------------------------------------------------------

function init()
{ for (var i=0;i<iMaxWindow;i++) aWinList[i]=null;
  myConnection = new ConnectionData();    // create a connection object
  // WindowLauncher();
  deskMgr  = new DeskManagerObject("deskMgr");
  debugMgr = new DebugObject('debugMgr');
  launchTimeAndUserDeamon();
  //setDebugLevel(2);
  launchSplashScreen();


  debugMgr.logMsg('<BR><a href="#"
onclick="callHelp(\'abcd.html\',\'ABCD\')">Call Help System</a><BR><BR>');
  debugMgr.logMsg('<BR>Set debugLevel to <a href="#"
onclick="setDebugLevel(0);">[0]</a> <a href="#"
onclick="setDebugLevel(1);">[1]</a> <a
href="#"onclick="setDebugLevel(2);">[2]</a>');
  debugMgr.logMsg('<BR>$RCSfile: akodium.html,v $');
  debugMgr.logMsg('<BR>$Revision: 1.42 $');
  debugMgr.logMsg('<BR>$Date: 2001/05/21 07:46:40 $');
  debugMgr.logMsg('<BR>$Name:  $');
  debugMgr.logMsg('<BR>$Author: burtin $');
  debugMgr.logMsg('<BR>The Knowledge DHTML (v2.0)');
}

// create the buttons in the upper  left corner
function SetButtonsPanel()
 {
    BDIV.innerHTML= button("close_button", "bt_close", "if(window.confirm('You
are going to leave The Knowledge Factory
!'))oHomePage.navigateHomePage();event.returnValue=false;", "Exit")+
                    button("Layout_button","bt_layout"  ,"saveAllWindows();"
,"Save layout")+
                    button("info_button"  ,"bt_info"
,"launchPersonalise(0,ixx,iyy,500,400,0,deskMgr.currentDesktop);" ,"Personalize")
+
                    button("search_button","bt_magn"    ,"launchSearch();"
,"Search") +
                    button("tree_button"  ,"bt_tree"
,"launchTeee(0,ixx,iyy,300,300,0,deskMgr.currentDesktop);"        ,"Tree") +
                    //button("target_button","bt_target"  ,""
,"Not implemented") +
                    button("iagent_button","bt_plus"
,"launchIagent(0,ixx,iyy,400,150,0,deskMgr.currentDesktop);"      ,"Add
document")+
                    button("recom_button","bt_recom"
,"launchRecommendations(0,ixx,iyy,300,150,0,deskMgr.currentDesktop);"
,"Recommendations")+
                    deskMgr.getHtml()+' '+
                    debugMgr.getHtml(); // add a hidden debug button, you may
remove it for debug
   setTimeout("info_button.src='pictures/desktop/bt_info_off.gif'",500); // makes
buttons appear (IE bug?)
 }


//   applet window launches, call them for creating windows

//   iRestore : if this is a restore
```

```
//   ix        : window's X
//      iy        : window's Y
//      iw        : window's width
//      ih        : window's height
//      ic        : window's state  (maximized, iconified ....)
//      idesk     : window's desk (-1 = all desks)

function launchPersonalise(iRestore,ix,iy,iw,ih,ic,idesk)
 {var  uniqId = 'PE';
   if (canIOpenWindow(uniqId))
    { var a=new
WindowObject(0,"akodi.client.personalise.personalise.class","applet","","personal
ise.jar,interface.jar,akodi.jar","1.0.0.0, 1.0.0.0,
1.0.0.0",ixx,iyy,deskMgr.currentDesktop,iw,ih,ic,null,"handle_info",false,0,"",nu
ll);
        if (a.bDivStatus>STATUS_NOTCREATED) { a.sUniqWinID = uniqId;} // all
desktops
      }
   }

function launchSearch() {openWS(0,'-1','1','0');}

function openWS(iRestore,sId,sTab,kId)
 { launchWs(iRestore,sId,sTab,kId,ixx,iyy,400,420,0, deskMgr.currentDesktop);}

function launchWs(iRestore,sId,sTab,kId,ix,iy,iw,ih,ic,idesk)
 { sParam      = new Array('ID',sId,'TAB',sTab,'KID',kId);
   Log('OPENWS '+iRestore+' '+sId+' '+sTab+' '+kId);
   var  uniqId = 'W'+sId;
   if (canIOpenWindow(uniqId))
    { var w = new
WindowObject(iRestore,"akodi.client.workspace.workspace.class","applet","pictures
/workspace","workspace.jar,interface.jar,akodi.jar","1.0.0.0, 1.0.0.0,
1.0.0.0",ixx,iyy,idesk,iw,ih,0,sParam,"handle_w",false,0,"",null);
      if (w.bDivStatus>STATUS_NOTCREATED)  w.sUniqWinID = uniqId ;
    }
 }

function openKO(iRestore,sRid, iTabToOpen)
 { var tab=0;
   if (iTabToOpen) tab=iTabToOpen;
   var sParam = new Array('TAB',tab);
   var sDoc =
myConnection.sServerProtocol+"://"+myConnection.sServerAddr+"/servlet/kobj?cmd=do
wnload&id="+sRid;
   Log("opening:"+sDoc);
   new
WindowObject(iRestore,"akodi.client.ko.KObjectViewer.class","applet","pictures/ko
bjectviewer","kobject.jar,interface.jar,akodi.jar","1.0.0.0, 1.0.0.0 ,
1.0.0.0",ixx,iyy,deskMgr.currentDesktop,500,400,0,sParam,"handle_k",true,54,sDoc,
sRid);
 }

function launchTeee(iRestore,ix,iy,iw,ih,ic,idesk)
 { Log('launch@ '+ix+' '+iy);
   new
WindowObject(iRestore,"akodi.client.tree.Tree.class","applet","pictures/tree","tr
ee.jar,interface.jar,akodi.jar","1.0.0.0, 1.0.0.0,
1.0.0.0",ix,iy,idesk,iw,ih,ic,null,"handle_t",false,0,"",null);
 }


function launchRecommendations(iRestore,ix,iy,iw,ih,ic,idesk)
 { var  uniqId = 'RE';
```

```
   if (canIOpenWindow(uniqId))
   { Log('launch@ '+ix+' '+iy);
     var a=new
WindowObject(iRestore,"akodi.client.recommendation.RecommendationViewer.class","a
pplet","pictures/recommendations","recommendation.jar,interface.jar,akodi.jar","1
.0.0.0, 1.0.0.0, 1.0.0.0",ix,iy,idesk,iw,ih,ic,null,"handle_r",false,0,"",null);
       if (a.bDivStatus>STATUS_NOTCREATED) { a.sUniqWinID = uniqId }
  }
 }

function launchAlert()
 { var  uniqId = 'AL';
   if (canIOpenWindow(uniqId))
    { var x=ixx; var y=iyy; // little hack to put the window at specific place
      ixx=iGiveMeDeskTopWidth()-360;  iyy=iGiveMeDeskTopHeight()-210;
      var a= new
WindowObject(0,"akodi.client.alert.AlertViewer.class","applet","pictures/alerts",
"alert.jar,interface.jar,akodi.jar,kobject.jar","1.0.0.0,1.0.0.0,1.0.0.0,1.0.0.0"
,ixx,iyy,deskMgr.currentDesktop,350,200,0,null,"handle_al",false,0,"",null);
       if (a.bDivStatus>STATUS_NOTCREATED) { a.sUniqWinID = uniqId }
       a.iDivDesk = -1;     // all desktops
      ixx=x; iyy=y;        // restore previous default position
     }
 }

function launchIagent(iRestore,ix,iy,iw,ih,ic,idesk)
 { var  uniqId = 'IA';
   if (canIOpenWindow(uniqId))
    { sDoc = myConnection.sServerProtocol+"://"+
myConnection.sServerAddr+"/tkf/upload.html" ;
      var a=new
WindowObject(0,"akodi.client.ia.InputAgent.class","applet","","inputagent.jar,int
erface.jar,akodi.jar","1.0.0.0, 1.0.0.0,
1.0.0.0",ix,iy,idesk,iw,ih,ic,null,"handle_+",true,30,sDoc,null);
      a.oKobjElement.navigate(sDoc);
       if (a.bDivStatus>STATUS_NOTCREATED) { a.sUniqWinID = uniqId;} // all
desktops
     }
  }

function launchKobj(iRestore,sRid, ix,iy,iw,ih,ic,idesk,iTabToOpen)
 { var tab=0;
   if (iTabToOpen) tab=iTabToOpen;
   var sParam = new Array('TAB',tab);
   var sDoc =
myConnection.sServerProtocol+"://"+myConnection.sServerAddr+"/servlet/kobj?cmd=do
wnload&id="+sRid;
   Log("opening:"+sDoc);
   var a = new
WindowObject(iRestore,"akodi.client.ko.KObjectViewer.class","applet","pictures/ko
bjectviewer","kobject.jar,interface.jar,akodi.jar","1.0.0.0, 1.0.0.0,
1.0.0.0",ixx,iyy,idesk,500,400,0,sParam,"handle_k",true,54,sDoc,sRid);
}

function launchCrashTest()
 { new
WindowObject(0,"Crashtest.class","applet","","",ixx,iyy,deskMgr.currentDesktop,10
0,100,0,null,"",false,0,"",null); }

function launchHelloSwing()
 { new
WindowObject(0,"HelloSwingApplet.class","applet","","",ixx,iyy,deskMgr.currentDes
ktop,130,25,0,null,"",false,0,"",null); }
```

```
// fill the Date & time Div
var sLastHeader="";
function SetTimeAndUser()
 { var sMonths =  new
Array('Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec');
   var date = new Date();
   var sDate =date.getDate()+' '+sMonths[date.getMonth()]+' '+date.getYear();
   var header=    '<font face="Sans-Serif" size=-1 color="#ffffff"><b
unselectable="on"> '+sDate;
   if  (myConnection.sUsername!='') header=header+' - '+myConnection.sUsername;
   header=header+'</b></font>     '    ;
   if (header!=sLastHeader) {LOGDIV.innerHTML = header; sLastHeader=header ; }

 }

function launchTimeAndUserDeamon()
{ SetTimeAndUser();
  setTimeout("launchTimeAndUserDeamon();",60000);
}

// ----------------------------------------------------------------------// Error
handler, the little bug window, launchError function is called
// for java applets, its content is dynamicaly created and stored in
// ErrorLog varaible. All errors are stacked (last error in first) in
// this variable.
//----------------------------------------------------------------------

var ErrorLog='';
function launchError(sError,sErrorTxt)
 {
   if (sError=='TKF-09001')
{setTimeout("window.top.location='end_session.html';",1500); return;} //09001 =
SESSION TERMINATED
   currentDate = new Date();
   var h = currentDate.getHours();
   var m = currentDate.getMinutes();
   var s = currentDate.getSeconds();
   if (m<10) m='0'+m;
   if (s<10) s='0'+s;
   var TKFERRORLOG= window.open('','TKFERRORLOG','width=450,height=250');
   ErrorLog='<DIV style="position:relative; width=100%;background-color:#B0B0B0;
layer-background-color:#B0B0B0 ;background: #B0B0B0">'+
                    '<font face="Arial,Helvetica" color=#000000
>'+h+':'+m+':'+s+' - <b>'+sError+'</b></font></div><BR>'+
                                '<DIV style="position:relative;left=10;">'+
                    '<font face="Arial,Helvetica" color=#000000
size=2>'+sErrorTxt+'</font></div><BR><BR>'+ErrorLog;

   var sendContent='<DIV id="SND"
style="position:absolute;width:450;height:195;top:55;left:0;z-
index:100;background-color:#808080; layer-background-color:#808080 ;background:
#808080;visibility:hidden">'+
                    '<FORM NAME="frmDescr"   Method=Post
OnReset="SND.style.visibility=\'hidden\'" '+
                                'OnSubmit="
document.frmDescr.action=\'mailto:bugs@akodi.com?cc=godot@akodi.com&subject=[bug
TKF] bug report\';SND.style.visibility=\'hidden\';return true;"
Enctype="text/plain">'+
                    'Description:<input type="text" Name="titre"
size="55"><br>'+
                    ' <font face="Arial,Helvetica" color=#FFFF00
size=-1>Please, explain us how you did that</font>:<br>'+
```

```
                                '<TEXTAREA  Name="Description" rows=5 cols=53
wrap="Yes"></TEXTAREA><br><br>'+
                                '</input>'+
                                '<div align=right><Input Type="submit"
value="Send"><Input Type="reset" value="Cancel">  </div><br>'+
                                '</FORM>'+
                                '</DIV>';

   var content='<DIV
style="position:absolute;width:450;height:30;top:0;left:0;background-
color:#DE1F4D; layer-background-color:#DE1F4D ;background: #DE1F4D">'+
                '<font face="Arial,Helvetica" color=#FFFFDC
size=+2><b>'+sError+'</b></font>'+
                                '<DIV DIV
style="position:absolute;width:75;height:30;top:0;left:375;"><a href="#"
onclick="SND.style.visibility=\'visible\';return true;"><img border=0
src="pictures/desktop/sendbug.gif" alt="send bug report to Akodi"></a></div>'+
                                '</DIV>'+
                                '<DIV
style="position:absolute;width:450;height:220;top:30;left:0;overflow:auto;backgro
und-color:#FFFFDC; layer-background-color:#FFFFDC ;background:
#FFFFDC"">'+ErrorLog+'</div>'+
          sendContent;
  TKFERRORLOG.document.write(content);
  TKFERRORLOG.focus();
 }

function button(sName,sfile,sAction,sAlt)
 { var sContent =  '<a href="#"  onfocus="blur();"'+
       ' onmouseover="'+sName+'.src=\'pictures/desktop/'+sfile+'_on.gif\';"'+
         ' onmouseout ="'+sName+'.src=\'pictures/desktop/'+sfile+'_off.gif\';"'+
         '
onmousedown="'+sName+'.src=\'pictures/desktop/'+sfile+'_down.gif\';"'+
       ' onmouseup  ="'+sName+'.src=\'pictures/desktop/'+sfile+'_on.gif\';"'+
       ' onclick="'+sAction+'"'+
       '><img name="'+sName+'"  border=0
src="pictures/desktop/'+sfile+'_off.gif\" alt="'+sAlt+'"></a> ';

  return      sContent;
 }

//-------------------------------------------------------------------//
DESKTOP MANAGER
// ie the little menu with the for desktops
//-------------------------------------------------------------------

function DeskManagerObject(sName)
 { this.currentDesktop      = 1;
   this.name                = sName;
   this.bVisible            = false;
   this.desktopOptionHtml   = DeskManagerObject_desktopOptionHtml;
   this.onMouseOver         = DeskManagerObject_onMouseOver;
   this.onMouseOut          = DeskManagerObject_onMouseOut;
   this.selectDesk          = DeskManagerObject_selectDesk;
   this.getHtml             = DeskManagerObject_getHtml;
   this.onMouseClick        = DeskManagerObject_onMouseClick;
   this.show                = DeskManagerObject_show;
   this.hide                = DeskManagerObject_hide;
 }
// return
function  DeskManagerObject_desktopOptionHtml(n)
 { var sContent =  '<a href="#"  onfocus="blur();"'+
       ' onmouseover="DESKBT'+n+'.src=\'pictures/desktop/mnu_d'+n+'_on.gif\';"'+
         ' onmouseout
="DESKBT'+n+'.src=\'pictures/desktop/mnu_d'+n+'_off.gif\';"'+
```

```
      ' onclick="'+this.name+'.selectDesk('+n+')"'+
      '><img name="DESKBT'+n+'"  border=0
src="pictures/desktop/mnu_d'+n+'_off.gif\" ></a><BR>';
   return    sContent;
}

function DeskManagerObject_getHtml()
 { var sContent=  '<a href="#"  onfocus="blur();"'+
                  'onmouseover="'+this.name+'.onMouseOver();" '+
                           'onmouseout ="'+this.name+'.onMouseOut();" '+
                           'onclick    ="'+this.name+'.onMouseClick();" '+
                           '><img name="DESKBT"  border=0
src="pictures/desktop/bt_d'+this.currentDesktop+'_off.gif\" ></a>'+
                           '<DIV  id="DESKOPTION"
style="position:absolute;top:25;left:217;height:100;width:70;z-
index:1000;visibility:hidden" UNSELECTABLE="on">';
   for (var i=1;i<5;i++) sContent=sContent+   this.desktopOptionHtml(i);

   sContent=sContent+'</div>';
   return sContent;
 }

function DeskManagerObject_onMouseOver()
{DESKBT.src='pictures/desktop/bt_d'+this.currentDesktop+'_on.gif';}
function DeskManagerObject_onMouseOut()
{DESKBT.src='pictures/desktop/bt_d'+this.currentDesktop+'_off.gif';}
function DeskManagerObject_show()
{DESKOPTION.style.visibility='visible';this.bVisible=true;}
function DeskManagerObject_hide()
{DESKOPTION.style.visibility='hidden';this.bVisible=false;}
function DeskManagerObject_onMouseClick(){if (this.bVisible) this.hide(); else
this.show();}

function DeskManagerObject_selectDesk(n)
{ if (this.currentDesktop!=n)
  { this.currentDesktop=n;
    DESKBT.src='pictures/desktop/bt_d'+this.currentDesktop+'_off.gif';
    this.hide();
    for (var i=0;i<iMaxWindow;i++)
      if (aWinList[i]!=null) aWinList[i].deskTopChanged(n);
    giveFocusToFrontWindow();
  }
}

//-----------------------------------------------------------------------
// HelpSystem
//-----------------------------------------------------------------------

function callHelp(sFilename,sAnchor)
 {var file = sFilename;
  if (file=='') file='book1.htm';   // the defaukt root document
  if (sAnchor!='') file=file+"#"+sAnchor;

  var sContent =
  '<frameset COLS="25%,*">'+
     '<frame SRC="help/toc.htm" NAME="TOC">'+
     '<frameset ROWS="*,25">'+
        '<frame SRC="help/'+file+'"   NAME="BODY">'+
        '<frame SRC="help/navbar.htm" NAME="NAV" frameborder="no" marginheight=0
marginwidth=10 scrolling="no" >'+
     '</frameset>'+
  '</frameset>';
  var TKFHELPWIN=
  window.open('','On_Line_Help','width=850,height=500,resizable=yes,titlebar=no');
```

```
TKFHELPWIN.document.write(sContent);
TKFHELPWIN.document.close();
TKFHELPWIN.focus();
}

//-------------------------------------------------------------------------//
DESKTOP MANAGER
//-------------------------------------------------------------------------

function DebugObject(sName)
{ this.msg          = '';
  this.name         = sName;
  this.visible      = false;
  this.logMsg       = DebugObject_logMsg;
  this.onMouseClick = DebugObject_onMouseClick;
  this.getHtml      = DebugObject_getHtml;
  this.show         = DebugObject_show;

}

function DebugObject_logMsg(stringMsg)
 { this.msg=stringMsg+'<BR>'+this.msg;
   DEBUGDIV.innerHTML="<font face='Arial' color=#FFFFFF size=-
1>"+this.msg+"</font>"+
   '<div align=center><B><font face="Arial" color="#007070" size=+2> AKODIUM
V2.0</font></B></div><BR>';

 }

function DebugObject_show()
 {this.visible=true;DEBUGDIV.style.visibility='visible';}

function DebugObject_onMouseClick()
 {this.visible=!this.visible;
  if (this.visible)
   { DEBUGDIV.style.visibility='visible';
     DEBUGBT.src='pictures/desktop/debug_on.gif';
   }
  else
  { DEBUGDIV.style.visibility='hidden';
    DEBUGBT.src='pictures/desktop/debug_off.gif';
  }
 }

function setDebugLevel(l) {iDebugLevel=l;Log('Debug Level is '+l);}
function DebugObject_getHtml()
 {  var sContent= '<a href="#"  onfocus="blur();"'+
                          'onclick    ="'+this.name+'.onMouseClick();" '+
                          '><img name="DEBUGBT"  border=0
src="pictures/desktop/debug_off.gif" alt="debug"></a>';
    return sContent;
 }

// save layout fct
function saveAllWindows() {
  SPLASHAPPLET.clearWinRef();
  if (iDebugLevel>0) Log('*** Terminating Session') ;
  for (var i=0;i<iMaxWindow;i++) {
    if  (aWinList[i]!=null) {
          if  (aWinList[i].isRunning()) {
        if (iDebugLevel>0)  Log('Saving for '+aWinList[i].sWinName+'
('+aWinList[i].className+')');
```

```
aWinList[i].oAppletElement.saveLayout(parseInt(aWinList[i].iDivDesk),parseInt(aWi
nList[i].getZindex()));
                }
        }
      }
      processForSaveLayout = true;
      alert('Layout saved.');
}

function document_onclick() {

      if(currentActivatedWindow != null) currentActivatedWindow.releaseFocus();

      currentActivatedWindow = null;

}

</SCRIPT>

</HEAD>

<BODY bgcolor="#008080" topmargin="0" leftmargin="0"  onload=init()
unselectable="on">
<IE:HOMEPAGE ID="oHomePage" />
<!-- Top black Div -->
<DIV  align=CENTER style="position:absolute;top:0;left:0;height:55;width:100%
;background-color:#000000;" unselectable="on"> </div>
<DIV  align=CENTER style="position:absolute;top:0;left:0;;width:100%;z-
index:100;" unselectable="on"><div align=center unselectable="on"> <img
src='pictures/desktop/tkf_original.gif' unselectable="on"></div> </div>
<!-- handles DIV -->
<DIV  style="position:absolute;top:34;left:0;height18;width:100% ; background-
color:#101010;clip: rect(0,3000,18,0);z-index:1;" unselectable="on"> </div>
<!-- orange DIV -->
<DIV  style="position:absolute;top:52;left:0;height:1;width:100% ;background-
color:#FF9C00;clip: rect(0,3000,1,0);z-index:2;" unselectable="on"> </div>
<!-- Date&UserDiv DIV -->
<DIV  id="LOGDIV" align="right"
style="position:absolute;top:9;left:9;height:22;width:100%;z-index:4;"
unselectable="on"></div>
<!-- button DIV -->
<DIV  id="BDIV" style="position:absolute;top:9;left:9;height:22;width:370;z-
index:101;" unselectable="on">
<font face='Arial' color=#FFFFFF unselectable="on">Loading layout, please
wait...</font>
</div>
<!-- DESKTOP DIV-->
<DIV id="DESKDIV" align=CENTER onclick="document_onclick()"
style="position:absolute;top:55;left:0;height:100%;width:100% ;background-
color:#008080;" unselectable="on"> </div>
<!-- DEBUG DIV -->
<DIV  id="DEBUGDIV"
style="position:absolute;top:150;left:50;height:500;width:400;z-
index:0;visibility:hidden;overflow:auto;background-color:#009090;border: 1px
solid #000000;" unselectable="on"></div>
</BODY>
</HTML>
```

Hereunder is a possible way of coding the Wincontainer applet.

```
/*************************************************************
 ** (c) AKODI SA - The Knowledge Factory
 ** $RCSfile: WinContainer.java,v $
 ** $Revision: 1.7 $
 ** $Date: 2001/05/21 14:58:01 $
 ** $Name:  $
 ** $Author: burtin $
 *************************************************************/
package akodi.client.cli;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;
import javax.swing.event.*;
import java.net.*; //for URL object3
import java.beans.*;

import akodi.tkf.error.*;
import akodi.util.tools.Log;
import akodi.util.xml.*;

import akodi.client.common.InterfaceConstants;
// LiveConnect... for JavaScript
import netscape.javascript.JSObject;


/**
 * WinContainer  is a container for all panel of TKF. It use for create a window
 * border with title bar and move, resize action in DTHML
 *
 * @author    burtin@akodi.com
 * @version   $Revision: 1.7 $
 */
public class WinContainer extends JApplet {
  public Log l=Log.getLog();
  boolean   isStandalone = false;            // set to FALSE, do not change it
  /**
   * The title bar of the applet window.
   */
      protected TitleBarWindow titlebar;

  /**
   * The name of the window in the html page.
   */
      protected String refWindow;
  /**
   * The CID of the applet window.
   */
      protected int cid;
  /**
   * The address of the server.
   */
  public String serverAddr;
  /**
   * The protocol.
   */
  public String serverProtocol;

  /**
   * The page html.
   */
      protected JSObject html = null;
```

```
/**
 * The listener for the resize and move applet window.
 */
    protected MouseInputAdapter       borderListener;
    protected MouseInputAdapter       glasspaneDispatcher;
protected PropertyChangeListener  wincontainerListener;

    // Component for the look and feel
    protected JComponent northPane;
    protected JComponent centerPane;

protected JPanel northBorderPanel;
protected JPanel southBorderPanel;
protected JPanel eastBorderPanel;
protected JPanel westBorderPanel;
protected JPanel separatorPanel;

    private Rectangle parentBounds = null;

    private boolean opened = true;

protected String windowState = null;

/**
 *  The applet window can be closed.
 * @see #closable
 */
    protected boolean closable = true;
private boolean confirmClosing = true;

/**
 * The applet window has been closed
 */
    protected boolean isClosed;

/**
 * The applet window can be expanded to the size of the desktop pane
 */
    protected boolean maximizable = true;

/**
 * The applet window has been expanded to its maximum size.
 * @see #maximizable
 */
    protected boolean isMaximum = false;

/**
 * The applet window can be restored.
 */
    protected boolean restorable = true;
/**
 * The applet window has been restored .
 * @see #restorable
 */
    protected boolean isRestored = true;

/**
 * The applet window can be iconable.
 */
    protected boolean iconable = true;
/**
 * The applet window has been iconized .
 * @see #restorable
 */
```

```
  protected boolean isIconized = false;
  protected boolean isIcon = false;

  /**
   * The applet window's size can be changed
   */
      protected boolean resizable = true;
  /**
   * The applet window is currently selected
   */
      protected boolean isSelected = true;

  /** The icon shown in the top-left corner of the frame
   */
      protected Icon frameIcon;
  /**
   * The title displayed in the applet window's title bar
   */
      protected String  title = "Untitle";

  protected Cursor currentCursor =
Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR);

// The border size of the applet window
  private int borderTop = 3;
  private int borderLeft = 3;
  private int borderBottom = 2;
  private int borderRight = 3;
      protected Insets borderSize = new Insets(borderTop, borderLeft,
borderBottom, borderRight);
      // The size to restore
      protected Rectangle restoreSize = null;

      private WinContainer winContainer;

  // some environnement
  private boolean ContainerRestrictedToAppletViewer = false;
  private boolean runningInDhtm                     = false;

  // Possible Look & Feels
  private String mac      = "ccm.sun.java.swing.plaf.mac.MacLookAndFeel";
  private String metal    = "javax.swing.plaf.metal.MetalLookAndFeel";
  private String motif    = "com.sun.java.swing.plaf.motif.MotifLookAndFeel";
  private String windows  = "com.sun.java.swing.plaf.windows.WindowsLookAndFeel";

  /** Bound property name. */
  public final static String CONTENT_PANE_PROPERTY = "contentPane";
  /** Bound property name. */
  public final static String MENU_BAR_PROPERTY = "menuBar";
  /** Bound property name. */
  public final static String TITLE_PROPERTY = "title";
  /** Bound property name. */
  public final static String LAYERED_PANE_PROPERTY = "layeredPane";
  /** Bound property name. */
  public final static String ROOT_PANE_PROPERTY = "rootPane";
  /** Bound property name. */
  public final static String GLASS_PANE_PROPERTY = "glassPane";
  /** Bound property name. */
  public final static String FRAME_ICON_PROPERTY = "frameIcon";

  /**
   * Constrained property name indicated that this frame has
   * selected status.
   */
```

```
public final static String IS_SELECTED_PROPERTY = "selected";
public final static String SELECTED_PROPERTY = "selectedByHtml";
/** Constrained property name indicating that the frame is closed. */
public final static String IS_CLOSED_PROPERTY = "closed";
/** Constrained property name indicating that the frame is maximized. */
public final static String IS_MAXIMUM_PROPERTY = "maximum";
/** Constrained property name indicating that the frame is iconified. */
public final static String IS_ICON_PROPERTY = "icon";

private int iDesk = 1;
private boolean isDrag = false;
private boolean isResize = false;
// The current Look & Feel
private String currentLookAndFeel = metal;

    public WinContainer() {
  1.setLevel(this.getClass(), Log.ERROR);
 }
     /**
      * Point d'entrée de l'applet.
      */
     public void installUI(boolean dev)
     {

ContainerRestrictedToAppletViewer = dev;

          this.installDefaults();
          this.installListeners();

          setNorthPane(createNorthPane(this));
this.getContentPane().add(this.northBorderPanel);
this.getContentPane().add(this.southBorderPanel);
this.getContentPane().add(this.eastBorderPanel);
this.getContentPane().add(this.westBorderPanel);
this.getContentPane().add(this.separatorPanel);

           this.parentBounds = this.getParent().getBounds();

// test for DHTML phase 1 : this to create a JSObject
if(!ContainerRestrictedToAppletViewer)
   try { html = JSObject.getWindow(this);}
catch (UnsatisfiedLinkError e) {ContainerRestrictedToAppletViewer=true;}
// test for DHTML phase 2 : try to call DHTML
if(!ContainerRestrictedToAppletViewer)
try {
  if(!ContainerRestrictedToAppletViewer) {
    if(!this.closable) {
      String args1[] = {this.refWindow, "0"};
      this.callHtmlMethod("setClosable",args1);
    }
    String args2[] = {};
    this.callHtmlMethod("amIRunningInDhtm",args2);
   }
   this.runningInDhtm=true;
  } catch (Exception ex) {this.runningInDhtm=false;}
}
     ////////////////////////////////////////////////////////////////
```

```
    protected void installDefaults() {
    try {
          UIManager.setLookAndFeel(currentLookAndFeel);
          SwingUtilities.updateComponentTreeUI(this);
        } catch (Exception ex) {
          this.showException(ex);
      }
    this.northBorderPanel = new JPanel();
    this.northBorderPanel.setBackground(InterfaceConstants.WINDOW_BORDER_COLOR);
    this.southBorderPanel = new JPanel();
    this.southBorderPanel.setBackground(InterfaceConstants.WINDOW_BORDER_COLOR);
    this.eastBorderPanel = new JPanel();
    this.eastBorderPanel.setBackground(InterfaceConstants.WINDOW_BORDER_COLOR);
    this.westBorderPanel = new JPanel();
    this.westBorderPanel.setBackground(InterfaceConstants.WINDOW_BORDER_COLOR);
    this.separatorPanel = new JPanel();
    this.separatorPanel.setBackground(InterfaceConstants.WINDOW_BORDER_COLOR);
      }

      protected void installListeners(){
            borderListener = createBorderListener();
    glasspaneDispatcher = createGlassPaneDispatcher();
    wincontainerListener = createWinContainerChangeListener();
    this.addPropertyChangeListener(wincontainerListener);
            this.getContentPane().setLayout(createLayoutManager());
            installMouseHandlers(this.northBorderPanel);
            installMouseHandlers(this.southBorderPanel);
            installMouseHandlers(this.eastBorderPanel);
            installMouseHandlers(this.westBorderPanel);
    this.getGlassPane().addMouseListener(glasspaneDispatcher);
    this.getGlassPane().addMouseMotionListener(glasspaneDispatcher);

      }
      public void setCid(int iCid){
            cid = iCid;
      }
      public int getCid(){
            return cid;
      }

      public void setRefWindow(String ref){
            refWindow = ref;
      }
      public String getRefWindow(){
            return refWindow;
      }

////////////////////////////////////////////////////////////////////


  public MouseInputAdapter getMouseHandle(){
    return borderListener;
  }

  public void iAmRunning()
      { if (!ContainerRestrictedToAppletViewer)
     { String args2[] = {refWindow};
       winContainer.callHtmlMethod("iAmRunning",args2);
     }
  }

  // test for DHTML presence
  public boolean isThisDhtml()   {  return this.runningInDhtm; }
```

```
/**
 * method to show exceptions, its call Dhtml which will raise an error window
 *
 * @param ex. the exception to log
 */
public void showTkfException(TkfException ex)
 {
  if (ContainerRestrictedToAppletViewer)
   { System.out.println(ex.toString());}
  else
   {String msg1 =ex.toString();
    String msg2 ="";
    for (int i=0;i<msg1.length();i++) // substitutes cariage returns by <BR>
     { if (msg1.charAt(i)==10) msg2=msg2+"<br>";
       else if (msg1.charAt(i)=='>') msg2=msg2+"&lg;";
       else if (msg1.charAt(i)=='<') msg2=msg2+"&lt;";
       else msg2=msg2+msg1.charAt(i);
     }
     if(!ContainerRestrictedToAppletViewer){
       String args2[] = {ex.getErrorCode(),msg2};
       winContainer.callHtmlMethod("launchError",args2);
     }
   }
 }

 public void showException(Exception ex)
 {
   if (ContainerRestrictedToAppletViewer)
    { ex.printStackTrace();}
   else
    {String msg1 =ex.toString();
     String msg2 ="";
     for (int i=0;i<msg1.length();i++) // substitutes cariage returns by <BR>
      { if (msg1.charAt(i)==10) msg2=msg2+"<br>";
        else if (msg1.charAt(i)=='>') msg2=msg2+"&lg;";
        else if (msg1.charAt(i)=='<') msg2=msg2+"&lt;";
        else msg2=msg2+msg1.charAt(i);
      }
      if(!ContainerRestrictedToAppletViewer){
        String args2[] = {"Java exception",msg2+"<br>A stack trace of the
error is reported in the Java Console"};
        winContainer.callHtmlMethod("launchError",args2);
      }
     ex.printStackTrace();
    }
  }


 /**
  * Calling this method with a value of <code>true</code> to close
  * the frame.
  *
  * @param b a boolean, where true means "close the frame"
  */
       public void setClosed(boolean b) {
              if (isClosed == b) {
                     return;
              }
              isClosed = b;
              if(isClosed) {
                     opened = false;
              }else if(!opened) {
                     opened = true;
              }
              if(isClosed == true) {
```

```
        closeWindow(this);
      }
        }

/**
 * Returns whether this WinContainer be closed by some user action.
 * @return true if the frame can be closed
 */
      public boolean isClosable() {
            return closable;
      }

 /**
  * Sets that the <code>WinContainer</code> can be maximized by
  * some user action.
  *
  * @param b a boolean  where true means the frame can be maximized
  */

   public void setMaximizable(boolean b)
  {  Boolean oldValue = maximizable ? Boolean.TRUE : Boolean.FALSE;
     Boolean newValue = b ? Boolean.TRUE : Boolean.FALSE;
     maximizable = b;
     firePropertyChange("maximizable", oldValue, newValue);
   }

 /**

  * Maximizes and restores the frame.  A maximized frame is resized to
  * fully fit the <code>AKODIUM</code> area associated with the
  * <code>WinContainer</code>.
  * A restored frame's size is set to the <code>WinContainer</code>'s
  * actual size.
  *
  * @param b  a boolean, where true maximizes the frame and false
  *           restores it
  */

 public void setMaximum(boolean b) {
   if (isMaximum == b) {
     return;
   }
   Boolean oldValue = isMaximum ? Boolean.TRUE : Boolean.FALSE;
   Boolean newValue = b ? Boolean.TRUE : Boolean.FALSE;
   /* setting isMaximum above the event firing means that
      property listeners that, for some reason, test it will
          get it wrong... See, for example, getNormalBounds() */
   isMaximum = b;

   if(isMaximum) { borderSize = new Insets(0, 0, 0, 0);  }
  else { borderSize = new Insets(borderTop, borderLeft, borderBottom,
borderRight);   }

   firePropertyChange(IS_MAXIMUM_PROPERTY, oldValue, newValue);
 }
      public boolean isMaximizable() {
            return maximizable;
      }

      public boolean isMaximum() {
            return isMaximum;
      }

      public boolean isRestorable() {
            return restorable;
```

```
        }

        public boolean isRestored() {
              return isRestored;
        }
    public void setRestore() {
      if(isRestored == false) {
        isMaximum = false;
        isRestored = true;
        isIconized = false;
        borderSize = new Insets(borderTop, borderLeft, borderBottom, borderRight);
                     restoreWindow(this);

      }
        }
        /**
         * Returns whether the applet window can be iconified by
         * som user action.
         *
         * @return true if the frame can be iconified
         */

        public boolean isIconifiable() {
              return iconable;
        }

    /**
     * Returns whether the <code>WinContainer</code> is currently
     * iconified.
     *
     * @return true if the frame is iconified
     */

    public boolean isIcon() {
      return isIcon;
    }

        /**
     * Iconizes and de-iconizes the frame.
     *
     * @param b a boolean, where true means to iconify the frame and
     *          false means to de-iconify it
     * @exception PropertyVetoException when the attempt to set the
     *            property is vetoed by the <code>WinContainer</code>
     *
     */

    public void setIcon(boolean b) {

      /*if (isIcon == b) {
        return;
      }*/

      Boolean oldValue = isIcon ? Boolean.TRUE : Boolean.FALSE;
      Boolean newValue = b ? Boolean.TRUE : Boolean.FALSE;
      isIcon = b;
      firePropertyChange(IS_ICON_PROPERTY, Boolean.FALSE, newValue);

    }

        /**
         * Sets that the <code>applet window</code> can be resized by some
         * user action.
         *
         * @param b  a boolean, where true means the frame can be resized
         */
        public void setResizable(boolean b) {
```

```
            resizable = b;
        }

        /**
         * Returns whether the <code>applet window</code> can be resized
         * by some user action.
         *
         * @return true if the frame can be resized
         */
        public boolean isResizable() {
            // don't allow resizing when maximized.
            return isMaximum ? false : resizable;
        }

        /**
         * Sets the WinContainer caption.
         * @see #getCaption
         *
         * @param caption  the String to display in the title bar
         */
    public void setCaption(String caption) {
            this.title = caption;
      this.titlebar.repaint();//paintComponent(this.titlebar.getGraphics());
      if(!ContainerRestrictedToAppletViewer){
        String args[] = {this.refWindow, caption};
        this.callHtmlMethod("setHandleTip", args);
      }
        }
    /**
     * Returns the title of the WinContainer.
     *
     * @return a String containing the WinContainer's title
     * @see #setCaption
     */
        public String getCaption() {
            return title;
        }

     /**
     * Selects and deselects the WinContainer.
     * A WinContainer normally draws its title bar differently if it is
     * the selected frame, which indicates to the user that this
     * internalFrame has the focus.
     *
     * @param selected  a boolean, where true means the frame is selected
     *                  (currently active) and false means it is not
     * @exception PropertyVetoException when the attempt to set the
     *            property is vetoed by the receiver.
     */

    public void setSelected(boolean selected) {
      if ((isSelected == selected) || (selected && !isShowing())) {
        return;
      }
      Boolean oldValue = isSelected ? Boolean.TRUE : Boolean.FALSE;
      Boolean newValue = selected ? Boolean.TRUE : Boolean.FALSE;
        /* will be not implemented at the moment:
       We don't want to leave focus in the previously selected
       frame, so we have to set it to *something* in case it
       doesn't get set in some other way (as if a user clicked on
       a component that doesn't request focus).  If this call is
       happening because the user clicked on a component that will
       want focus, then it will get transfered there later.
       We test for parent.isShowing() above, because AWT throws a
       NPE if you try to request focus on a lightweight before its
```

```
        parent has been made visible */

            /*if(selected) {
         //will be not implemented at the moment
              JRootPane r = getRootPane();
          if (r.getCurrentFocusOwner() != null) { do nothing
              }else if(r.getPreviousFocusOwner() != null) {
                r.getPreviousFocusOwner().requestFocus();
              }else {
                getContentPane().requestFocus();
              }
          getContentPane().requestFocus();
           . }*/
        if(selected) requestFocus();
        isSelected = selected;
        firePropertyChange(IS_SELECTED_PROPERTY, oldValue, newValue);
        repaint();
      }

    public void setSelectedByHtml() {
       isSelected = true;
       firePropertyChange(SELECTED_PROPERTY, Boolean.FALSE, Boolean.TRUE);
     }
     /**
      * Returns whether the applet window is the currently "selected" or
      * active frame.
      *
      * @return true if the frame is currently selected (active)
      * @see #setSelected
      */

          public boolean isSelected() {
                return isSelected;
          }

          /*
          * Adds necessary mouseHandlers to currentPane and adds it to
          * applet.
          * Reverse process for the newPane.
          */
          protected void replacePane(JComponent currentPane, JComponent newPane) {
                if(currentPane != null) {
                      deinstallMouseHandlers(currentPane);
                      this.getContentPane().remove(currentPane);
                }
                if(newPane != null) {
          installMouseHandlers(newPane);
                      this.getContentPane().add(newPane);
                }
           }

        // mouse handler
        // install - deinstall

           protected void deinstallMouseHandlers(JComponent c) {
                 c.removeMouseListener(borderListener);
                 c.removeMouseMotionListener(borderListener);
           }

           protected void installMouseHandlers(JComponent c) {
                 c.addMouseListener(borderListener);
                 c.addMouseMotionListener(borderListener);
           }
```

```
    ////////////////////////////////////////////////////////////////////     //
    // listener border

        protected MouseInputAdapter createBorderListener() {
            return new BorderListener();
        }

        protected PropertyChangeListener createWinContainerChangeListener() {
            return new WinContainerPropertyChangeListener();

        }

    ////////////////////////////////////////////////////////////////////////
    // dispatcher
    // glasspane

        protected MouseInputAdapter createGlassPaneDispatcher() {
            return new GlassPaneDispatcher();
        }

    ////////////////////////////////////////////////////////////////////////
    //
    // create layoutmanager
    //
        protected LayoutManager createLayoutManager(){
            return new WinContainerLayout();
        }


    ////////////////////////////////////////////////////////////////////////
    //
    // create border pane
    // north , center

        protected JComponent createNorthPane(WinContainer winContainer){
            this.titlebar = new TitleBarWindow(winContainer);
            return titlebar;
        }

        protected JComponent createCenterPane(JApplet winContainer){
            return new JPanel();
        }

    ////////////////////////////////////////////////////////////////////////
    //
    // set get pane
    // north south west east

        public JComponent getNorthPane(){
            return northPane;
        }
        public void setNorthPane(JComponent c){
            replacePane(northPane, c);
            northPane = c;
        }
        public JComponent getCenterPane(){
            return centerPane;
        }
        public void setCenterPane(JComponent c){
            replacePane(centerPane, c);
            centerPane = c;
        }

      public void init() {
```

```
switch(Integer.parseInt(getParameter("DEBUGLEVEL","0"))) {
  case 0:
    l.setLevel(this.getClass(), Log.ERROR);
    break;
  case 1:
    l.setLevel(this.getClass(), Log.WARNING);
    break;
  case 2:
    l.setLevel(this.getClass(), Log.DEBUG);
    break;
}
this.setEnabled(false);
this.winContainer = this;
this.setCid(Integer.parseInt( getParameter("cid","0") ));
this.iDesk = Integer.parseInt( getParameter("DESK","1") );
this.serverAddr = getParameter("serverAddr", "127.0.0.1:8080");
this.serverProtocol = getParameter("serverProtocol", "http");
        this.setRefWindow( getParameter("winname") );
this.restoreSize = new Rectangle();
this.restoreSize.x = Integer.parseInt(getParameter("APPLETLEFT","0"));
this.restoreSize.y = Integer.parseInt(getParameter("APPLETTOP","0"));
this.restoreSize.height = Integer.parseInt(getParameter("APPLETHEIGHT","0"));
this.restoreSize.width = Integer.parseInt(getParameter("APPLETWIDTH","0"));
int state = Integer.parseInt(getParameter("APPLETSTATE","0"));
switch(state) {
  case 0:
    isIconized = false;
    isMaximum = false;
    isRestored = true;
    break;
  case 1:
    isIconized = true;
    isMaximum = false;
    isRestored = false;
    break;
  case 2:
    isIconized = false;
    isMaximum = true;
    isRestored = false;
    break;
  case 3:
    isIconized = true;
    isMaximum = true;
    isRestored = false;
    break;
  default:
    isIconized = false;
    isMaximum = false;
    isRestored = true;
    break;
}
this.installUI(ContainerRestrictedToAppletViewer);
}

public void start() {
  this.setEnabled(true);
  this.setVisible(true);
}

public void stop() {
  this.setEnabled(false);
  this.setVisible(false);
  super.stop();
}
```

```
  public void destroy() {
    super.destroy();
  }

  public void installContentPane(JComponent c) {
    setCenterPane(c);
  }
      public Dimension getPreferredSize()    {
            return getContentPane().getPreferredSize();
      }
      public Dimension getMinimumSize()  {
            return getContentPane().getMinimumSize();
      }
      public Dimension getMaximumSize() {
            return new Dimension(Integer.MAX_VALUE, Integer.MAX_VALUE);
      }

//////////////////////////////////////////////////////////////////////
/// Border Listener Class
//////////////////////////////////////////////////////////////////////

//**
// * Listens for border adjustments.
*/

protected class BorderListener extends MouseInputAdapter
                              implements SwingConstants {
            // _x & _y are the mousePressed location in absolute coordinate
system
            int _x, _y;
            // __x & __y are the mousePressed location in source view's
coordinate system
            int __x, __y;

            Rectangle startingBounds;
            int resizeDir;
            protected final int RESIZE_NONE  = 0;
            int resizeCornerSize = 16;

            public void mouseClicked(MouseEvent e) {
                  if(e.getClickCount() > 1 && e.getSource() ==
winContainer.titlebar) {
                   if(winContainer.isMaximizable()) {
                              if(!winContainer.isMaximum())
                                    winContainer.setMaximum(true);
                              else
                                    winContainer.setMaximum(false);
                        }
                  }
            }

            public void mouseReleased(MouseEvent e) {
      String args0[] = {winContainer.refWindow};

      if((/*e.getSource() == winContainer.northBorderPanel |
          e.getSource() == winContainer.southBorderPanel |
          e.getSource() == winContainer.westBorderPanel |
          e.getSource() == winContainer.eastBorderPanel |
          e.getSource() == winContainer.titlebar |*/
          isDrag) &&
      (!winContainer.isMaximum())) {
        // get the coordinate of the top-left corner in html page
        if(!ContainerRestrictedToAppletViewer){
            Integer Top = (Integer)
winContainer.callHtmlMethod("iObjectTop",args0);
```

```
                restoreSize.y = Top.intValue();
                  Integer Left =(Integer)
winContainer.callHtmlMethod("iObjectLeft",args0);
                restoreSize.x = Left.intValue();
        }
        // get the width and height of the container
        restoreSize.width = winContainer.getBounds().width;
        restoreSize.height = winContainer.getBounds().height;
      }

      // reinit all to zero
                _x = 0;
                _y = 0;
                __x = 0;
                __y = 0;

      isDrag = false;
      isResize = false;
                startingBounds = null;
                resizeDir = RESIZE_NONE;
            }

            public void mousePressed(MouseEvent e) {
       if(!winContainer.isSelected()) {
                        winContainer.setSelected(true);
                }
       Point p = SwingUtilities.convertPoint((Component)e.getSource(),e.getX(),
e.getY(), null);
                __x = e.getX();
                __y = e.getY();
                _x = p.x;
                _y = p.y;
                startingBounds = winContainer.getBounds();
       startingBounds.x = restoreSize.x;
       startingBounds.y = restoreSize.y;

       if(e.getSource() == titlebar) isDrag = true;

                if(!winContainer.isResizable() || e.getSource() == titlebar)
{
                        resizeDir = RESIZE_NONE;
                        return;
                }

       if(e.getSource() == winContainer.northBorderPanel) {
         if(e.getX() < resizeCornerSize) {
                        resizeDir = NORTH_WEST;
         }else if(e.getX() > winContainer.getWidth()                    - resizeCornerSize)
{
           resizeDir = NORTH_EAST;
                        }else {
                                resizeDir = NORTH;

         }
         isResize = true;
       }else if(e.getSource() == winContainer.southBorderPanel) {
         if(e.getX() < resizeCornerSize)
                        resizeDir = SOUTH_WEST;
                else if(e.getX() > winContainer.getWidth()
                                                - resizeCornerSize
)
                        resizeDir = SOUTH_EAST;
                else
                        resizeDir = SOUTH;
```

```
          isResize = true;
        }else if(e.getSource() == winContainer.westBorderPanel) {
          if(e.getY() < resizeCornerSize)
            resizeDir = NORTH_WEST;
          else if(e.getY() > winContainer.getHeight()- resizeCornerSize )
                                resizeDir = SOUTH_WEST;
                          else
                                resizeDir = WEST;
          isResize = true;
        }else if(e.getSource() == winContainer.eastBorderPanel ) {
                          if(e.getY() < resizeCornerSize) {
                                resizeDir = NORTH_EAST;
          }else if(e.getY() > winContainer.getHeight() - resizeCornerSize)
                                resizeDir = SOUTH_EAST;
                          else
                                resizeDir = EAST;
          isResize = true;
        }
              }

              public void mouseDragged(MouseEvent e) {
                    if ( startingBounds == null ) {
                          return;
                    }

        // use for calculate the move of the window
        int newX, newY, newW, newH;
        int deltaX;
                int deltaY;
                Dimension min;
                Dimension max;

                    //get point on screen
                Point p = SwingUtilities.convertPoint((Component)e.getSource(),
e.getX(), e.getY(), null);
                    // Handle a MOVE
                    if(isDrag) {
                          if (winContainer.isMaximum()) {
                                return;  // don't allow moving of maximized
frames.
                          }
                          newX = startingBounds.x - (_x - p.x);
                          newY = startingBounds.y - (_y - p.y);

                          if(!ContainerRestrictedToAppletViewer){
                            String args1[] = {winContainer.refWindow,
Integer.toString(newX), Integer.toString(newY)};
                            winContainer.callHtmlMethod("moveObject",args1);

            // need this only for tempo
            String args0[] = {winContainer.refWindow};
            winContainer.callHtmlMethod("amIRunningInDhtm",args0);
          }
                          return;
                    }

                    if(!winContainer.isResizable()) {
                          return;
                    }

        // Handle a RESIZE
                    if(isResize) {

                      deltaX = _x - p.x;
```

```
                deltaY = _y - p.y;

               newX = startingBounds.x;
               newY = startingBounds.y;
               newW = startingBounds.width;
                newH = startingBounds.height;

               switch(resizeDir) {
               case RESIZE_NONE:
                     return;
                  case NORTH:
                     if(startingBounds.height + deltaY <
getMinimumSize().height)
                           deltaY = -(startingBounds.height -
getMinimumSize().height);
                     else if(startingBounds.height + deltaY >
getMaximumSize().height)
                           deltaY = (startingBounds.height -
getMinimumSize().height);
                     newX = startingBounds.x;
                     newY = startingBounds.y - deltaY;
                        newW = startingBounds.width;
                          newH = startingBounds.height + deltaY;
                          break;
                     case NORTH_EAST:
                        if(startingBounds.height + deltaY <
getMinimumSize().height)
                              deltaY = -(startingBounds.height -
getMinimumSize().height);
                        else if(startingBounds.height + deltaY >
getMaximumSize().height)
                           deltaY = (startingBounds.height -
getMinimumSize().height);
                        if(startingBounds.width - deltaX <
getMinimumSize().width)
                           deltaX = (startingBounds.width -
getMinimumSize().width);
                        else if(startingBounds.width - deltaX >
getMaximumSize().width)
                              deltaX = -(startingBounds.width -
getMinimumSize().width);
                        newX = startingBounds.x;
                        newY = startingBounds.y - deltaY;
                           newW = startingBounds.width - deltaX;
                           newH = startingBounds.height + deltaY;
                           break;
                  case EAST:
                     if(startingBounds.width - deltaX <
getMinimumSize().width)
                           deltaX = (startingBounds.width -
getMinimumSize().width);
                     else if(startingBounds.width - deltaX >
getMaximumSize().width)
                           deltaX = -(startingBounds.width -
getMinimumSize().width);
                     newW = startingBounds.width - deltaX;
                        newH = startingBounds.height;
                        break;
                     case SOUTH_EAST:
                        if(startingBounds.width - deltaX <
getMinimumSize().width)
```

```
                                    deltaX = (startingBounds.width -
getMinimumSize().width);
                                else if(startingBounds.width - deltaX >
getMaximumSize().width)
                deltaX = -(startingBounds.width - getMinimumSize().width);

                            if(startingBounds.height - deltaY <
getMinimumSize().height)
                                    deltaY = (startingBounds.height -
getMinimumSize().height);
                            else if(startingBounds.height - deltaY >
getMaximumSize().height)
                                    deltaY = -(startingBounds.height -
getMinimumSize().height);

                              newW = startingBounds.width - deltaX;
                        newH = startingBounds.height - deltaY;
                              break;
                          case SOUTH:
                              if(startingBounds.height - deltaY <
getMinimumSize().height)
                                    deltaY = (startingBounds.height -
getMinimumSize().height);
                                else if(startingBounds.height - deltaY >
getMaximumSize().height)
                                    deltaY = -(startingBounds.height -
getMinimumSize().height);

                            newW = startingBounds.width;
                            newH = startingBounds.height - deltaY;
                            break;
                          case SOUTH_WEST:
                              if(startingBounds.height - deltaY <
getMinimumSize().height)
                                    deltaY = (startingBounds.height -
getMinimumSize().height);
                                else if(startingBounds.height - deltaY >
getMaximumSize().height)
                                    deltaY = -(startingBounds.height -
getMinimumSize().height);

                            if(startingBounds.width + deltaX <
getMinimumSize().width)
                                    deltaX = -(startingBounds.width -
getMinimumSize().width);
                            else if(startingBounds.width + deltaX >
getMaximumSize().width)
                                    deltaX = (startingBounds.width -
getMinimumSize().width);

                            newX = startingBounds.x - deltaX;
                            newY = startingBounds.y;
                            newW = startingBounds.width + deltaX;
                            newH = startingBounds.height - deltaY;
                            break;
                        case WEST:
                            if(startingBounds.width + deltaX <
getMinimumSize().width)
                                    deltaX = -(startingBounds.width -
getMinimumSize().width);
                              else if(startingBounds.width + deltaX >
getMaximumSize().width)
                                    deltaX = (startingBounds.width -
getMinimumSize().width);
```

```
                        newX = startingBounds.x - deltaX;
                        newY = startingBounds.y;
                        newW = startingBounds.width + deltaX;
                        newH = startingBounds.height;
                            break;
                    case NORTH_WEST:
                        if(startingBounds.width + deltaX <
getMinimumSize().width)
                            deltaX = -(startingBounds.width -
getMinimumSize().width);
                        else if(startingBounds.width + deltaX >
getMaximumSize().width)
                            deltaX = (startingBounds.width -
getMinimumSize().width);
                        if(startingBounds.height + deltaY <
getMinimumSize().height)
                            deltaY = -(startingBounds.height -
getMinimumSize().height);
                        else if(startingBounds.height + deltaY >
getMaximumSize().height)
                            deltaY = (startingBounds.height -
getMinimumSize().height);
                        newX = startingBounds.x - deltaX;
                        newY = startingBounds.y - deltaY;
                        newW = startingBounds.width + deltaX;
                        newH = startingBounds.height + deltaY;
                            break;
                    default:
                        return;
                }
            if(!ContainerRestrictedToAppletViewer){
                String args1[] = {winContainer.refWindow,
Integer.toString(newX), Integer.toString(newY), Integer.toString(newW),
Integer.toString(newH)};
                winContainer.callHtmlMethod("resizeObject",args1);
            }
        }
    }
    public void mouseMoved(MouseEvent e)    {
        if(!winContainer.isResizable()) {
            return;
        }
        if(e.getSource() == winContainer.northBorderPanel) {
            if(e.getX() < resizeCornerSize )
winContainer.setCursor(Cursor.getPredefinedCursor(Cursor.NW_RESIZE_CURSOR));
            else if(e.getX() > winContainer.getWidth() - resizeCornerSize)
winContainer.setCursor(Cursor.getPredefinedCursor(Cursor.NE_RESIZE_CURSOR));
            else
winContainer.setCursor(Cursor.getPredefinedCursor(Cursor.N_RESIZE_CURSOR));
            return;
        }else if(e.getSource() == winContainer.southBorderPanel) {
            if(e.getX() < resizeCornerSize)
winContainer.setCursor(Cursor.getPredefinedCursor(Cursor.SW_RESIZE_CURSOR));
            else if(e.getX() > winContainer.getWidth() - resizeCornerSize)
```

```
winContainer.setCursor(Cursor.getPredefinedCursor(Cursor.SE_RESIZE_CURSOR));
        else
winContainer.setCursor(Cursor.getPredefinedCursor(Cursor.S_RESIZE_CURSOR));
        return;
      }else if(e.getSource() == winContainer.westBorderPanel) {
        if(e.getY() < resizeCornerSize)
winContainer.setCursor(Cursor.getPredefinedCursor(Cursor.NW_RESIZE_CURSOR));
                        else if(e.getY() > winContainer.getHeight() -
resizeCornerSize )
winContainer.setCursor(Cursor.getPredefinedCursor(Cursor.SW_RESIZE_CURSOR));
        else
winContainer.setCursor(Cursor.getPredefinedCursor(Cursor.W_RESIZE_CURSOR));
      }else if(e.getSource() == winContainer.eastBorderPanel ) {
        if(e.getY() < resizeCornerSize)
winContainer.setCursor(Cursor.getPredefinedCursor(Cursor.NE_RESIZE_CURSOR));
        else if(e.getY() > winContainer.getHeight() - resizeCornerSize)
winContainer.setCursor(Cursor.getPredefinedCursor(Cursor.SE_RESIZE_CURSOR));
                        else
winContainer.setCursor(Cursor.getPredefinedCursor(Cursor.E_RESIZE_CURSOR));
        return;
                    }else {
winContainer.setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR));
      }
            }

            public void mouseEntered(MouseEvent e) {}

            public void mouseExited(MouseEvent e) {
                winContainer.setCursor(Cursor.getDefaultCursor());
      winContainer.doOnLeaved();
            }
       };
////////////  End BorderListener Class  ////////////////////////

//////////////////////////////////////////////////////////////////////////
/// WinContainer Property Change Listener Class
//////////////////////////////////////////////////////////////////////////
/**
 * Listens for border adjustments.
*/

 public class WinContainerPropertyChangeListener implements
PropertyChangeListener {
/** Detects changes in state from the WinContainer and handles actions.*/

    public void propertyChange(PropertyChangeEvent evt) {
      String prop = (String)evt.getPropertyName();

      WinContainer f = (WinContainer)evt.getSource();
      Object newValue = evt.getNewValue();
      Object oldValue = evt.getOldValue();
      if(WinContainer.IS_CLOSED_PROPERTY.equals(prop)) {
        closeWindow(f);
          }else if(WinContainer.IS_MAXIMUM_PROPERTY.equals(prop)) {
```

```
            if(newValue == Boolean.TRUE) {
            maximizeWindow(WinContainer.this);
          }else {
            restoreWindow(WinContainer.this);
            }
          titlebar.setButtonIcons();
                titlebar.enableActions();
           }else if(WinContainer.IS_ICON_PROPERTY.equals(prop)) {
          if(newValue == Boolean.TRUE) {
           iconifyWindow(WinContainer.this);
            }/*else {
           deiconifyWindow(WinContainer.this);
            }*/
          }else if(WinContainer.IS_SELECTED_PROPERTY.equals(prop)) {
            if(newValue == Boolean.TRUE && oldValue == Boolean.FALSE) {
           activateWindow(f);
                  winContainer.getGlassPane().setVisible(false);
          }else if(newValue == Boolean.FALSE && oldValue == Boolean.TRUE) {
            //deactivateWindow(f);
              winContainer.getGlassPane().setVisible(true);
          }
            }else if(WinContainer.SELECTED_PROPERTY.equals(prop)) {
              if(newValue == Boolean.TRUE && oldValue == Boolean.FALSE) {
                    winContainer.getGlassPane().setVisible(false);
          }else if(newValue == Boolean.FALSE && oldValue == Boolean.TRUE) {
            winContainer.getGlassPane().setVisible(true);
          }
            }
        }
      }
////////////  End WinContainerPropertyChangeListener Class  ///////////////

////////////////////////////////////////////////////////////////////////////
/// GlassPaneDispatcher Listener Class
////////////////////////////////////////////////////////////////////////////

    /**
     * Listens for border adjustments.
     */

    protected class GlassPaneDispatcher extends MouseInputAdapter
                                        implements SwingConstants {

      private Component mouseEventTarget = null;
              public void mouseMoved(MouseEvent e) {
        if(borderListener != null) {
          borderListener.mouseMoved(e);
        }
        forwardMouseEvent(e);
      }

      public void mouseDragged(MouseEvent e) {
        if(borderListener != null) {
          borderListener.mouseDragged(e);
        }
      }

      public void mouseClicked(MouseEvent e) {}

      public void mouseEntered(MouseEvent e) {
        forwardMouseEvent(e);
      }

      public void mouseExited(MouseEvent e) {
        forwardMouseEvent(e);
      }
```

```
    public void mousePressed(MouseEvent e) {
      if(borderListener != null) {
        borderListener.mousePressed(e);
      }
      forwardMouseEvent(e);
    }

    public void mouseReleased(MouseEvent e) {
      if(borderListener != null) {
        borderListener.mouseReleased(e);
      }
      forwardMouseEvent(e);
      isDrag = false;
      isResize = false;
    }

    /*
     * Forward a mouse event to the current mouse target, setting it
     * if necessary.
     */
    private void forwardMouseEvent(MouseEvent e) {
      Component target =
findComponentAt(winContainer.getRootPane().getLayeredPane(),
                                          e.getX(), e.getY());
      if(target == titlebar) {
        isDrag = true;
        return;
      }else {
        isDrag = false;
      }

      if(target == winContainer.northBorderPanel |
         target == winContainer.southBorderPanel |
         target == winContainer.westBorderPanel |
         target == winContainer.eastBorderPanel) {
        isResize = true;
      }else {
        isResize = false;
      }

      if (target != mouseEventTarget) {
        setMouseTarget(target, e);
      }
      retargetMouseEvent(e.getID(), e);
    }

    /*
     * Find the lightweight child component which corresponds to the
     * specified location.  This is similar to the new 1.2 API in
     * Container, but we need to run on 1.1.  The other changes are
     * due to Container.findComponentAt's use of package-private data.
     */
    private Component findComponentAt(Container c, int x, int y) {
      if (!c.contains(x, y)) {
            return c;
      }
      int ncomponents = c.getComponentCount();
      Component component[] = c.getComponents();
      for (int i = 0 ; i < ncomponents ; i++) {
          Component comp = component[i];
          Point loc = comp.getLocation();
          if ((comp != null) && (comp.contains(x - loc.x, y - loc.y)) &&
(comp.isDisplayable()) &&
            (comp.isVisible() == true)) {
```

```
            // found a component that intersects the point, see if there
            // is a deeper possibility.
            if (comp instanceof Container) {
              Container child = (Container) comp;
              Point childLoc = child.getLocation();
              Component deeper = findComponentAt(child,
                                    x - childLoc.x, y - childLoc.y);

            if (deeper != null) {
                    return deeper;

            }
              }else {
                return comp;
              }
             }
        }
        return c;
    }

      /*
       * Set the child component to which events are forwarded, and
       * synthesize the appropriate mouseEntered and mouseExited events.
       */
      private void setMouseTarget(Component target, MouseEvent e) {
        if (mouseEventTarget != null) {
              retargetMouseEvent(MouseEvent.MOUSE_EXITED, e);
        }
        mouseEventTarget = target;
        if (mouseEventTarget != null) {
              retargetMouseEvent(MouseEvent.MOUSE_ENTERED, e);
        }
       }

       /*
        * Dispatch an event clone, retargeted for the current mouse target.
        */
       void retargetMouseEvent(int id, MouseEvent e) {
         // fix for bug #4202966 -- hania
         // When retargetting a mouse event, we need to translate
         // the event's coordinates relative to the target.
         Point p = SwingUtilities.convertPoint(winContainer.getLayeredPane(),
                                               e.getX(), e.getY(),
                                               mouseEventTarget);

         MouseEvent retargeted = new MouseEvent(mouseEventTarget,
                                                id,
                                                e.getWhen(),
                                                e.getModifiers(),
                                                p.x,
                                                p.y,
                                                e.getClickCount(),
                                                e.isPopupTrigger());

          mouseEventTarget.dispatchEvent(retargeted);
       }
         };
      //////////  End GlassPaneDispatcher Class  ////////////////////////////

      ////////////////////////////////////////////////////////////////////////
      /// WinContainer Layout Class
      ////////////////////////////////////////////////////////////////////////

            public class WinContainerLayout implements LayoutManager {
                  public void addLayoutComponent(String name, Component c) {}
                  public void removeLayoutComponent(Component c) {}
                  public Dimension preferredLayoutSize(Container c)  {
                        Dimension result;
```

```
                        result = winContainer.getRootPane().getPreferredSize();
                        result.width += borderSize.left + borderSize.right;
                        result.height += borderSize.top + borderSize.bottom;

                        if(titlebar != null) {
                                Dimension d = titlebar.getPreferredSize();
                                result.width = Math.max(d.width, result.width);
                                result.height += d.height;
                        }
                        return result;
                }

                public Dimension minimumLayoutSize(Container c) {
                        Dimension result = new Dimension(0,0);

                        result.width += borderSize.left + borderSize.right;
                        result.height += borderSize.top + borderSize.bottom;

                        if(titlebar != null) {
                                Dimension d = titlebar.getMinimumSize();
                                result.width = Math.max(d.width, result.width);
                                result.height += d.height;
                        }

        if(getCenterPane() != null) {
                                Dimension d = getCenterPane().getMinimumSize();
                                result.width = Math.max(d.width, result.width);
                                result.height += d.height;
                        }
                        return result;
                }

                public void layoutContainer(Container c) {
                        int cx, cy, cw, ch;

                        cx = borderSize.left;
                        cy = borderSize.top;
                        cw = winContainer.getWidth() - borderSize.left -
borderSize.right;
                        ch = winContainer.getHeight() - borderSize.top -
borderSize.bottom;
                        if(northBorderPanel != null) {
          northBorderPanel.setBounds(0,0,winContainer.getWidth(),borderSize.top);
        }

        if(southBorderPanel != null) {
          southBorderPanel.setBounds(0,winContainer.getHeight() -
borderSize.top,winContainer.getWidth(),borderSize.top);
        }

        if(eastBorderPanel != null) {
          eastBorderPanel.setBounds(winContainer.getWidth()-
borderSize.right,borderSize.top,borderSize.right,winContainer.getHeight() -
borderSize.top);
        }

        if(westBorderPanel != null) {

westBorderPanel.setBounds(0,borderSize.top,borderSize.left,winContainer.getHeight
() - borderSize.top);
        }

                        if(titlebar != null) {
                                Dimension size = titlebar.getPreferredSize();
```

```
                    titlebar.setBounds(cx, cy, cw, size.height);
                    cy += size.height;
                    ch -= size.height;
                }

            if(separatorPanel != null) {
        separatorPanel.setBounds(borderSize.left,cy,winContainer.getWidth()-
borderSize.right,2);
        }
                if(getCenterPane() != null) {
                    getCenterPane().setBounds(cx, cy + 2, cw, ch -
borderSize.top);
                }
            }
        }
////////////// end of interframelayout //////////////////////

// used to get icon file
//   - if debbuging on, gets the file in current directory
//   - if debugging off, gets it at specified URL
//
 public ImageIcon getIconFile(String filename, String url)
     {  if (!isThisDhtml()) return new ImageIcon(filename);
        else return new ImageIcon(getURL(url+'/'+filename));
     }

 public ImageIcon getIconFile(String filename) {
   return new ImageIcon(ClassLoader.getSystemResource(filename));
 }

   /**
    * Constructs a new instance.
    */
   /**
    * getParameter
    * @param key
    * @param def
    * @return java.lang.String
    */
   public String getParameter(String key, String def) {
     if (isStandalone) {
       return System.getProperty(key, def);
     }
     if (getParameter(key) != null) {
       return getParameter(key);
     }
     return def;
   }

   /*
    *  make an URL  from a filename, u must use that way of doing
    *  for accessing files from an applet running in a browser.
    */
   protected URL getURL(String filename) {
     URL codeBase = getCodeBase();
     URL url = null;
     try {
       url = new URL(codeBase, filename);
     }catch (java.net.MalformedURLException e) {
       System.err.println("Couldn't create image: " +
                          "badly specified URL");

       return null;
     }
```

```
    return url;
   }


  public String createBlockXmlContainer(int type, String url, int ressourceId,
int iDesk, int iZindex, boolean layout) {
    XmlBuffer xml = new XmlBuffer();
    xml.doElement(XmlConst.CONTAINER);
    xml.doElement("width");xml.appendAttribute("value",
restoreSize.width);xml.done();
    xml.doElement("height");xml.appendAttribute("value",
restoreSize.height);xml.done();
    xml.doElement("left");xml.appendAttribute("value", restoreSize.x);xml.done();
    xml.doElement("top");xml.appendAttribute("value", restoreSize.y);xml.done();
    xml.doElement("zindex");xml.appendAttribute("value", iZindex);xml.done();
    xml.doElement("desk");xml.appendAttribute("value", this.iDesk);xml.done();
    xml.doElement("bopen");
    if(this.opened)
      xml.appendAttribute("value", "y");
    else
      xml.appendAttribute("value", "n");
    xml.done();
    if(layout) {
      xml.doElement("brestore");xml.appendAttribute("value", "y");xml.done();
    }

    xml.doElement("state");
    int i = 0;
    if(this.isIconized)
      i = i | 1;
    if(this.isMaximum)
      i = i | 2;
    xml.appendAttribute("value", i);
    xml.done();
    xml.doElement("type");xml.appendAttribute("value", type);       xml.done();
    xml.doElement("url"); xml.appendAttribute("value", url);        xml.done();
    xml.doElement("rid"); xml.appendAttribute("value", ressourceId);xml.done();
    xml.done();
    return xml.toString();
  }

//////////////////////////////////////////////////////////////////////////
//
// Method for dialog with DHTML
//
//////////////////////////////////////////////////////////////////////////

  /**
   * Calling this method with a string to call a method in the html page
   *
   * @param methodName a string, the name of the method to call
   * @param args[] a table of string, the params of the method to call
   *
   */

  public synchronized Object callHtmlMethod(String methodName, String args[]){
    return html.call(methodName, args);
  }

// This method is called when the wincontainer is closing
// By default is nothing, override this method for particular process

like InputAgent
  public void doOnClosing(){}
```

```
// This method is called when the mouse is exiting the applet
// By default is nothing, override this method for particular process like
InputAgent
  public void doOnLeaved(){}

// This method is called when the user wants to close the winContainer window.
    protected void closeWindow(WinContainer a) {
   doOnClosing();
   if(confirmClosing) {
     if(!ContainerRestrictedToAppletViewer){
       String args[] = {a.refWindow};
       a.callHtmlMethod("closeObject",args);
     }
   }else {
     isClosed = false;
     confirmClosing = true;
   }
    }

public void confirmClosing(boolean confirm) {
  this.confirmClosing = confirm;
}

    // This method is called when the winContainer window has to close because
// of an error.
    protected void closeWindowOnError(WinContainer a) {
  this.runningInDhtm=false;
  if(!ContainerRestrictedToAppletViewer){
    String args[] = {a.refWindow};
    a.callHtmlMethod("closeObject",args);
  }
    }

// This method is called when the user wants to maximize the winContainer window.
protected void maximizeWindow(WinContainer a) {
   if(!ContainerRestrictedToAppletViewer){
              String args[] = {a.refWindow};
              a.callHtmlMethod("maximizeObject",args);
   }
    }

  public void reSize(Dimension dim) {
     if(!ContainerRestrictedToAppletViewer){
        String args1[] = {this.refWindow,
Integer.toString(restoreSize.x),Integer.toString(restoreSize.y),Integer.toString(
dim.width), Integer.toString(dim.height)};
              this.callHtmlMethod("resizeObject",args1);

     }
  }

  // This method is called when the user wants to restore the winContainer window.
     protected void restoreWindow(WinContainer a) {
     if(!ContainerRestrictedToAppletViewer){
              String args1[] = {a.refWindow, Integer.toString(restoreSize.x),
Integer.toString(restoreSize.y), Integer.toString(restoreSize.width),
Integer.toString(restoreSize.height)};
              a.callHtmlMethod("resizeObject",args1);

     }
      }

  // This method is called when the user wants to iconify the winContainer window.
     protected void iconifyWindow(WinContainer a) {
    if(!ContainerRestrictedToAppletViewer){
```

```
                String args1[] = {a.refWindow};
                a.callHtmlMethod("invertState",args1);
    }
      }
 /** This will activate <b>f</b> moving it to the front. It will
  * set the current active frame (if any) IS_SELECTED_PROPERTY to false.
  * There can be only one active frame across all Layers.
  */

  public void activateWindow(WinContainer a) {
    if(!ContainerRestrictedToAppletViewer){
                String args1[] = {a.refWindow};
                a.callHtmlMethod("activateWindow",args1);
    }
  }

  // implements javax.swing.DesktopManager
  public void deactivateWindow(WinContainer a) {}
}
```

It will be appreciated by persons skilled in the art that the present invention is not limited by what has been particularly shown and described hereinabove. Rather the present invention may include combinations and sub-combinations of the various features disclosed as well as modifications and extensions thereof which fall under the scope of the following claims.

Claims

1. A method for presenting information within independent windows displayed in a single HTML page characterized in that it comprises the following steps,
   - downloading a HTML page containing JavaScript methods from a web server;
   - displaying a desktop page at user's screen, this desktop page comprising a working area for the setting up of DHTML divisions within which independent WebWindows applets may be executed,
   - setting up of a window manager, by execution of JavaScript methods embedded in the HTML desktop page, that provides the functions for creating, displaying and managing said WebWindows applets
   - setting up by the window manager of a logical bi-directional communication path that allows synchronization between WebWindows applets and the DHTML divisions of the desktop page.
   - accepting user's input either in the desktop page or in the WebWindows applets,
   - synchronizing the WebWindows applets with the DHTML divisions in the desktop page

2. A method according to claim 1, characterized in that the window manager allows the creation, opening, closing, resizing, and positioning of independent web windows within the desktop area of the desktop page.

3. A method according to one of the preceding claims characterized in that each of the WebWindows applet are implemented as Java applets

having a data structure derived from a common parent WinContainer itself derived from the JApplet class.

4. A method according to one of the preceding claims characterized in that the desktop communicate with a Web server using the HTTP protocol for downloading images, content or applets and in that the WebWindow applets can use any protocol to communicate with the web server.

5. A method according to one of the preceding claims, characterized in that the desktop page communicates with the individual WebWindows applets through Java method invocation the WebWindows applets communicate with the desktop page thanks to JSObject calls.

6. A graphical user interface comprising a single DHTML desktop page downloaded from a web server, this desktop page having JavaScript methods for implementing a window manager that allows the creation and manipulation of independent WebWindows applets executing each in an independent dynamic HTML division.

7. A graphical user interface according to claim 6, characterized in that the desktop page comprises a title bar having buttons and menus for WebWindows management functions as well as a working area for displaying and arranging WebWindows applets.

8. A graphical user interface according to claim 6 or 7, characterized in that the title bar in the desktop page comprises icons indicating the status of the WebWindows applets.

9. Use of DHTML divisions and Java applets within a standard HTML page downloaded from a Web server for implementing a windows manager that allows the user to execute applets or to display information in separate independent windows displayed in HTML divisions of the downloaded HTML page.

US 1078350802P1

Creation date: 03-03-2004
Indexing Officer: SGEBREHIWOT - SARA GEBREHIWOT
Team: OIPEScanning
Dossier: 10783508

Legal Date: 02-20-2004

No.	Doccode	Number of pages
1	FOR	20
2	FOR	72
3	FOR	56

Total number of pages: 148

Remarks:

Order of re-scan issued on .....